ELSEVIER

Discrete Applied Mathematics 116 (2002) 193–229

DISCRETE APPLIED MATHEMATICS

The vehicle routing problem with pickups and deliveries on some special graphs☆

Tali Eilam Tzoreff[1], Daniel Granot*, Frieda Granot, Greys Sošić

Faculty of Commerce and Business Administration, University of British Columbia, 2053 Main Mall, Vancouver, BC, Canada V6T 1Z2

Received 3 March 1999; revised 22 May 2000; accepted 19 June 2000

Dedicated to Peter L. Hammer on the occasion of his 65th birthday

Abstract

The vehicle routing with pickups and deliveries (VRPD) problem is defined over a graph $G=(V,E)$. Some vertices in G represent delivery customers who expect deliveries from a depot, and other vertices in G represent pickup customers who have available supply to be picked up and transported to a depot. The objective is to find a minimum length tour for a capacitated vehicle, which starts at a depot and travels in G while satisfying all the requests by the delivery and pickup customers, without violating the vehicle capacity constraint, and returns to a depot. We study the VRPD problem on some special graphs, including trees, cycles and warehouse graphs when the depots are both exogenously and endogenously determined. Specifically, we develop linear time algorithms for the VRPD problem on tree graphs and polynomial algorithms on cycle and warehouse graphs.

MSC: 68Q25; 05C05; 05C38; 90B10

Keywords: Vehicle routing; Polynomial algorithm; Tree; Cycle graph; Warehouse graph

1. Introduction

In this paper we study a vehicle routing problem defined on a graph $G=(V,E)$, with vertex set V and edge set E. One, or perhaps two vertices in G are depots, or distribution centres, while other vertices in G represent customers. Some of these customers, referred to as delivery or demand customers, require a shipment from a depot and others, referred to as pickup or supply customers, have some supply which they

☆ The authors gratefully acknowledge helpful comments by an anonymous referee on an earlier version of this paper. Research was partially supported by Natural Sciences and Engineering Research Council grants 3998 and 4181, by UBC HSS grants, and by a UBC Graduate Fellowship.

* Corresponding author. Tel.: +1-604-822-8432; fax: +1-604-822-9574.

E-mail address: daniel.granot@commerce.ubc.ca (D. Granot).

[1] Current address: Compugen Ltd., Tel-Aviv, Israel.

0166-218X/02/$ - see front matter
PII: S0166-218X(00)00283-3

would like to send to a depot. Items to be picked up at the supply vertices are distinct from those which have to be delivered to the demand vertices. The objective is to find a minimum length tour for a capacitated vehicle, which starts at a depot loaded with enough supply to satisfy the demand of customers, travels in G while delivering supply to the demand customers and collecting supply from the supply customers without violating the vehicle capacity constraint, and returns to a depot. We will refer to the above routing problem as the vehicle routing with pickups and deliveries (VRPD) problem.

Routing problems with pickups and deliveries have many real life applications, see, e.g., [3,4] and references cited therein. For example, it is reported by Casco et al. [4] that the US grocery industry has saved over $160 million a year in distribution costs since 1982, by allowing vehicles on their delivery routes to pickup large volume from suppliers. Other applications studied in the literature include pickup and delivery from quality stores [12], pickup and delivery for ocean-borne transportation and pickup and delivery of inner-city under-privileged children to summer vacations at volunteer families living out of town [8].

Chalasani et al. [6] (see also [5]) have studied a related problem which arises in industrial automation. Therein, parts arrive on a conveyer belt and have to be grasped by a robot's arm and then delivered to specified delivery points for packing or other processing. The objective is to minimize the total distance traveled by the robot's arm. When the belt is static, the problem is referred to as the k-Delivery TSP problem. When the robot's arm has an unbounded capacity, the only restriction for a tour to be feasible is that at any stage the number of parts picked up, by that stage, is at least equal to the number of parts delivered so far. Chalasani et al. [6] have developed a linear time algorithm for the k-Delivery TSP problem on a tree graph when k is unbounded, which was used to obtain a 2-approximation for this problem on a general graph. In the next section we show that the k-Delivery TSP problem with unbounded k is a special case of the VRPD problem. Thus, all the results obtained in this paper for the VRPD problem are also valid for the k-Delivery TSP problem with unbounded k.

A reduction from TSP can be used to show that the VRPD problem is, in general, NP-hard.[1] Indeed, for a given TSP problem, consider a corresponding VRPD problem, where a supply-vertex and a demand-vertex, both for a single unit, are placed very close to each point of the TSP problem. The vehicle is fully loaded, and the single depot coincides with the travelling salesman starting vertex. Clearly, an optimal solution to the VRPD problem in this instance is an optimal solution to the TSP as well. Thus, the VRPD problem is, in general, NP-hard, and most of the literature on the VRPD problem was devoted to the development and study of heuristics for solving this problem. One such heuristic, developed by Mosheiov [8], follows the tour of a good heuristic for the associated unconstrained vehicle routing problem, but starts from that point on the tour from which the pickups and deliveries can be carried out without violating the vehicle capacity constraint. A theoretically superior heuristic, based on a minimum

[1] For a similar reduction from TSP to the k-Delivery TSP problem, see Chalasani and Motwani [5].

spanning tree for the underlying graph, was developed by Anily and Mosheiov [2]. The objective function value for their heuristic is at most twice the optimal value of the VRPD problem. Mosheiov [10] has developed several heuristics for the VRPD problem with multiple vehicles, and in Mosheiov [9], heuristics for the optimal location of a single depot in a VRPD problem in which the customers' demands are not deterministic was developed and tested.

Guan [7], see also references cited therein, studied a related capacitated vehicle routing problem, defined over a path, tree and cycle graphs, which arises in a motion planning of a robot. Therein, supply which arise at some vertices has to be moved to some other vertices in the graph, so there are no central depots from which supply is either delivered or dispersed. Moreover, by contrast with the problem studied by Chalasani et al. [6], the items which have to be routed in the graph are distinct. A similar problem, referred to as the swapping problem, was earlier studied by Anily and Hassin [1].

In this paper we study the VRPD problem on some special graphs with one or possible two depots, whose location(s) is either exogenously or endogenously determined. Specifically, let $|V|$ denote the number of vertices in the graph. Then, we develop $O(|V|)$ algorithms for a VRPD problem defined on a tree graph, $O(|V|^2)$ and $O(|V|^2 \log |V|)$ algorithms for a VRPD problem defined over a cycle graph, and a polynomial algorithm for a VRPD problem defined on a warehouse graph, in which the number of aisles is not part of the input.

2. Preliminary and notation

Let $G = (V, E)$ be an undirected graph with edge length function $l : E \to R_+$ and request function $r : V \to Z$. A vertex v for which $r(v) > 0$ is referred to as a supply or pickup vertex, and if $r(v) < 0$ then v is referred to as a demand or delivery vertex. A vehicle, $N = N(c^1, c^2)$, with a maximum capacity c^1 and initial load c^2, is available at a depot $s \in V$. The vehicle routing with pickups and deliveries (VRPD) problem is concerned with finding a shortest tour for the vehicle which starts at s, travels along G while picking up supply from supply vertices and delivering supply to demand vertices without violating the vehicle capacity constraint, and ends the tour at $t \in V$. Throughout this paper we let c, $c = c^1 - c^2$, denote the available capacity on the vehicle before any pickups or deliveries.

It is assumed that the items to be picked up at the supply vertices are distinct from those delivered to the demand vertices, so that we cannot pickup an item from one vertex and deliver it to another vertex at the same trip. Further, we assume no preemption. That is, any item which is picked up at a supply vertex can only be unloaded at t, and any item loaded at s for delivery at a demand vertex v, can only be unloaded at v.

A tour T in the VRPD problem is denoted by the sequence of vertices visited by the vehicle with the corresponding requests actually performed in each one of them. That is,

$T=\{(x_0=s,b_0),(x_1,b_1),\ldots,(x_l=t,b_l)\}$, where $x_i \in V$, $(x_i,x_{i+1}) \in E$, and $|b_i|$ denotes the number of units the vehicle either picked up or delivered at x_i. If $b_i < 0$ (resp. $b_i > 0$), a delivery (resp. pickup) was performed at vertex x_i. Given a tour T and $(x_i,b_i) \in T$, we denote by $T_{x_i}=\{(x_0=s,b_0),(x_1,b_1),\ldots,(x_i,b_i)\}$ the prefix of T up to and including x_i. A *feasible tour* of the vehicle $N=N(c^1,c^2)$, starts at s and ends at t after fulfilling all the requests associated with all the vertices in V without violating the vehicle's capacity constraint. An *optimal tour* is a feasible tour of minimum total length.

In order to assure the existence of a feasible tour in G, we can assume that the total demand at all demand vertices is equal to c^2, and the total supply at all supply vertices is not larger than c^1.

The following is an important property of feasible tours for the VRPD problem defined over a general graph G.

Observation 1. *Tour feasibility is maintained when deliveries are advanced or pickups are postponed.*

In other words, suppose a feasible tour T visits a vertex v more than once. If v is a demand vertex, then T can deliver all the requested items on its first visit to v without affecting the tour feasibility. If v is a supply vertex, then T can pickup all the supply available at v on its last visit to v without affecting the tour feasibility. Actually, such changes to a feasible tour T may enable one to construct a shorter feasible tour therefrom. Accordingly, we have:

Assumption 2. *In the sequel, unless otherwise stated, we restrict ourselves to feasible tours that deliver all the demand on their first visit to a demand vertex and pickup all the available supply on their last visit to a supply vertex. For a single exception to this assumption see Observation* 6.

Assumption 2 implies that if (x_i,b_i), $(x_j,b_j) \in T$ for $x_i=x_j=v$ and a feasible tour T, then at most one of b_i and b_j is equal to $r(v)$, and the others are equal to zero. It also implies that if $(x_i,b_i) \in T$ with $b_i > 0$, then x_i is not visited again during the rest of the tour. That is, the rest of the tour will not be affected by the deletion of vertex x_i from G. Accordingly, we have:

Corollary 3. *Given a feasible tour T and a supply vertex x_i. Immediately after T picks up the supply items from x_i, x_i can be deleted from G.*

For a VRPD(G,r,N,s,t) problem with predetermined s and t, a feasible tour T, and a vertex x, such that $(x,b) \in T$ *we associate a related VRPD*$(G_{T_x},r_{T_x},N_{T_x},x,t)$ *problem* defined as follows:

(i) T_x is the prefix of T up to x.
(ii) The *related graph*, G_{T_x}, is obtained from G by deleting all the vertices of G from which all the supply has been picked up by T_x.

(iii) The *related request function*, r_{T_x}, is obtained from r by setting to zero the request at every vertex v_i in G to which T_x delivered all the supply, while leaving the requests at all other vertices unchanged.

(iv) In the *related vehicle*, $N_{T_x} = (c^1_{T_x}, c^2_{T_x})$, $c^1_{T_x}$ is equal to c^1, and $c^2_{T_x}$ is equal to c^2 minus the number of items delivered by T_x plus the number of items picked up by T_x.

(v) The starting point, s, in the related problem is set to be $\hat{x}$, the vertex visited by T immediately after x, and the final point of the tour remains t.

Note that the remainder of the tour T, $T \setminus T_x$, is a feasible tour for the related problem VRPD $(G_{T_x}, r_{T_x}, N_{T_x}, x, t)$ from x to t. Moreover:

Proposition 4. *Let $T=\{(x_0,b_0),\ldots,(x_k,b_k)\}$ be an optimal tour for VRPD(G,r,N,s,t) with starting and ending vertices s and t. Suppose the first pick up by T is at x_j. That is, $(x_j,b_j) = \min_i\{(x_i,b_i) \in T,\ b_i > 0\}$. Then, $T \setminus T_{x_j}$ is an optimal tour for the VRPD $(G \setminus x_j, r_{T_{x_j}}, N_{T_{x_j}}, x_{j+1}, t)$ problem from x_{j+1} to t.*

Proof. Clearly, $T \setminus T_{x_j}$ is feasible for the related VRPD$(G_{T_{x_j}}, r_{T_{x_j}}, N_{T_{x_j}}, x_{j+1}, t)$ problem from x_{j+1} to t. Furthermore, $T \setminus T_{x_j}$ must be an optimal tour for the related problem. Otherwise, the tour consisting of T_{x_j} and an optimal tour to the related problem is feasible for VRPD (G,r,N,s,t) and shorter than T. □

Finally, let us consider the unbounded Delivery TSP problem on a graph G, studied by Chalasani et al. [6]. It can be formulated as follows. There are n distinct sources and n distinct sinks in a graph G, n identical parts, with one part at each source, and a robot's arm of unbounded capacity. The objective is to compute a minimum length route for the robot's arm to deliver exactly one part to each sink. Since the robot's arm has an unbounded capacity, the only restriction for a tour to be feasible, is that at any sink the number of parts picked up, up to that stage, is strictly larger than the number of parts delivered so far.

Apparently, the unbounded Delivery TSP problem can be formulated as a VRPD problem on the same graph G with n demand vertices and n supply vertices, where both the demand and supply are for a single item. Formally, given an unbounded Delivery TSP problem on a graph G with n sources, $s_1,\ldots,s_n$ and n sinks, $t_1,\ldots,t_n$, a starting point s and an ending point t, we define an equivalent VRPD problem on the same graph as follows. Let $r(s_1) = \cdots = r(s_n) = -1$ and $r(t_1) = \cdots = r(t_n) = +1$. That is, the source vertices in the unbounded-Delivery TSP problem are turned into demand vertices in the VRPD problem, while the sinks are turned into supply vertices. The vehicle is of capacity n and it starts fully loaded from the same starting point s and ends at the same ending point t. There is a one to one correspondence between the $s-t$ feasible tours for the unbounded-Delivery TSP problem and its equivalent VRPD problem, where two corresponding feasible tours are of the same length. This implies that an algorithm for finding an optimal tour for the VRPD problem on a graph G yields an algorithm for finding an optimal tour for the unbounded-Delivery

TSP problem. Thus, all the results obtained and algorithms developed in the sequel are equally valid for the unbounded-Delivery TSP problem.

3. The VRPD problem on a path

We analyze in this section the VRPD problem on a path L, when s and t are the two end vertices of L. This special case is used subsequently to develop algorithms for the VRPD problem on tree and cycle graphs. The case where the locations of s and t are either fixed at some arbitrary vertices on the line, or are endogenously determined, are left to Section 4, where the tree case is studied.

We use the following notation and definitions in this section. The path L has vertex set $V = \{v_0, v_1, \ldots, v_{n+1}\}$ and edge set $E = \{\bigcup_{i=0}^{n} e_i = (v_i, v_{i+1})\}$. Thus, v_0 and v_{n+1} are the leaves of L. We assume that $s = v_0$, $t = v_{n+1}$ and recall that we denote by c, $c = c^1 - c^2$, the available capacity of the vehicle starting at s. Let $S_i = \sum_{j=0}^{i} r(v_j)$ denote the cumulative requirements on L from v_0 to v_i. The partial sums S_i induce a partition of the path L to maximal subintervals on which the partial sums S_i strictly exceed the available capacity c, and maximal subintervals on which these partial sums do not strictly exceed c. We will refer to the former as *positive subintervals* and the latter as *non-positive subintervals.*

For a positive subinterval, I, we denote by $\check{v}(I)$ the lowest indexed vertex in I and by $\hat{v}(I)$ the vertex immediately following the maximal indexed vertex in I. Thus, if $I = \{v_p, \ldots, v_q\}$, then $\check{v}(I) = v_p$ and $\hat{v}(I) = v_{q+1}$. We will refer to $\hat{v}(I)$ as the *upper boundary vertex* of I. Observe that an upper boundary vertex must be a demand vertex.

A linear time algorithm for the VRPD problem on a path is presented below.

An algorithm for the path

Step 1: Computer the partial sums S_i, $i = 1, \ldots, n$

Step 2: While traversing L from s to t do:

(i) If v_i is a demand vertex (i.e., $r(v_i) < 0$), unload the required demand at v_i.

(ii) If v_i is a supply vertex (i.e., $r(v_i) > 0$) and $S_i \leqslant c$ (i.e., v_i is contained in a non-positive subinterval), load the supply available at v_i.

(iii) If v_i is a supply vertex and $S_i > c$ (i.e., v_i is contained in a positive subinterval), and this is the first visit of the vehicle at v_i, do not load the supply at v_i.

(iv) If v_i is an upper boundary vertex $\hat{v}(I)$ of a positive subinterval I do the following:
- Unload the demand required by $\hat{v}(I)$.
- Return idly to $\check{v}(I)$.
- Traverse the subinterval from $\check{v}(I)$ to $\hat{v}(I)$ while loading all the supply at supply vertices on I.
- Proceed from v_i to v_{i+1}.

The optimal tour, T, generated by the above algorithm can be equivalently described as follows. T starts at s and proceeds towards t. Deliveries are made at every demand

vertex the first time the vehicle visits such a vertex. If a supply vertex, v, is reached, no supply was left behind to be picked up and the vehicle has enough capacity, it picks up all the supply at v. If the vehicle does not have enough capacity to load the supply at v, it proceeds towards t leaving the supply behind. Once some supply was left behind, the vehicle continues to proceed towards t without picking up any supply, until T first reaches a vertex u at which it has enough capacity to pick up *all* the supply left behind. It then returns idly to the supply vertex v closest to s that has available supply, and then moves back towards u while picking up all supply between v and u.

The length of the optimal tour, $w(T)$, generated by the above algorithm is given by $w(T) = \sum (l(e_i)\colon S_i \leqslant c) + 3\sum (l(e_i)\colon S_i > c)$, where $e_i = (v_i, v_{i+1})$.

In the correctness proof of the algorithm we use the following lemma.

Lemma 5. *An optimal tour T for the VRPD(L, r, N, v_0, v_{n+1}) problem from $s = v_0$ to $t = v_{n+1}$ traverses every edge in L either once or three times.*

Proof. Every edge in L is traversed by T at least once. Suppose that there exists an edge e in L traversed by T more than once. Then, e is traversed by T at least three times. Suppose, on the contrary, that e is traversed by T more than three times. We will show that this implies the existence of a shorter feasible tour T', that traverses e exactly three times, contradicting the optimality of T. Let v_j–$v_{j'}$ be the maximal (with respect to inclusion) subpath of L containing e that is traversed more than once. That is, every edge in v_j–$v_{j'}$ is traversed more than once (though they do not have to be traversed the same number of times), and the edges (v_{j-1}, v_j), $(v_{j'}, v_{j'+1})$ are traversed exactly once. This implies that T first performs all the requests in the subpath s–v_{j-1}, then moves to v_j and after fulfilling all the requests in the subpath v_j–$v_{j'}$, it continues and performs the requests in the subpath $v_{j'+1}$–t. The following tour T' traverses v_j–$v_{j'}$ exactly three times, and is shorter than T. It coincides with T until T first reaches v_j. Then T' performs all the requests in the subpath v_j–$v_{j'}$ in the following manner: It first travels from v_j to $v_{j'}$ delivering supply to demand vertices while ignoring any pickup requests. Then, it moves back idly from $v_{j'}$ to v_j, and finally it returns from v_j to $v_{j'}$ performing all the pickup requests along this subpath. At this point T' has performed all the requests done by T, just before it moved from $v_{j'}$ to $v_{j'+1}$. Note that after moving from $v_{j'}$ to $v_{j'+1}$, both T and T' fulfilled all the requests in the subpath s–$v_{j'}$. From $v_{j'+1}$, T' coincides with T. Thus, T' is a feasible tour which coincides with T on the subpaths s–v_{j-1} and $v_{j'+1}$–t, and is shorter than T on the subpath v_j–$v_{j'}$, contradicting the optimality of T. □

Correctness proof of the algorithm. The sequence of partial sums, S_i, denotes the difference between the total supply and the total demand of vertices in the subpath s–v_i. When S_i does not strictly exceed c, the initial available capacity of the vehicle, the vehicle can perform all the supply and demand requests in the subpath s–v_i without leaving this subpath, i.e., without traversing edge $e_i = (v_i, v_{i+1})$. That is, when $S_i \leqslant c$,

there exists a feasible tour only for the subpath s–v_i. When $S_i > c$, there is no feasible tour just for the subpath s–v_i. Indeed, any feasible tour for L, including an optimal tour, traverses an edge e_i with $S_i > c$ strictly more than once. It will traverse it for the first time to unload some items at demand vertices on the subpath v_{i+1}–t, and then it traverses it back on the way to load excess supply left behind on the subpath s–v_i. Lemma 5 implies that an optimal tour traverses an edge e_i with $S_i > c$ exactly three times. The tour T generated by the algorithm is obviously feasible. Moreover, it is optimal since it traverses every edge e_i with $S_i > c$ exactly three times, and every other edge exactly once. Thus, the length, $w(T)$, of an optimal tour, T, generated by the algorithm is given by $w(T) = \sum (l(e_i): S_i \leqslant c) + 3 \sum (l(e_i): S_i > c)$. □

Finally, we note that if we relax Assumption 2, other optimal tours can be generated. Indeed:

Observation 6. *Different loading and unloading policies at supply and demand vertices could lead to different optimal tours, whose lengths, of course, are all equal. Indeed, the reader can verify that the following modification of the algorithm would also yield an optimal tour for a VRPD problem on a path. The vehicle proceeds from s towards t. Upon encountering a vertex v then*:

- *If* $r(v) < 0$ *and no supply was left behind, a delivery at v must be made.*
- *If* $r(v) < 0$ *and some supply was left behind, the delivery at v is optimal. However, if delivery is not done, then the net supply left behind is decreased by* $|r(v)|$.
- *If* $r(v) > 0$ *and there is not enough capacity on the vehicle, the supply at v is left behind.*
- *If* $r(v) > 0$ *and there is enough capacity on the vehicle, loading the supply at v is optional. However, if supply is not loaded, then net supply left behind is adjusted.*
- *Immediately when the vehicle reaches a vertex u at which it has enough capacity to pick up all net supply left behind, it returns to the supply vertex, w, closest to s at which supply was left behind. On its way from u to w, deliveries, if any, are made, and on the way back from w to u, all supply is picked up.*

4. The VRPD problem on a tree graph

In this section we study the VRPD problem on a tree graph. We start with the case where $s = t$. Then, we discuss the case where s and t are predetermined but not necessarily coincide, and finally we consider the case where either s or t or both are endogenously determined.

4.1. The case where $s = t$

Any feasible tour for the VRPD problem on a tree that starts and ends at the same vertex and visits all the vertices therein, traverses each edge at least twice. A linear time

algorithm which produces a feasible tour that traverses every edge exactly twice, i.e. an optimal tour, was first developed by Anily and Mosheiov [2] and later by Chalasani et al. [6], who used it to obtain a 2-approximation algorithm for the VRPD problem on a general graph. For completeness, we describe below this algorithm. Proofs of correctness can be found in [2,6].

Since the length of an optimal tour on a tree is independent of the choice of s, the vertex s can be chosen arbitrarily. Once chosen, we regard the tree as rooted away from s. For each x, $x \neq s$, we denote by $f(x)$ the father of x in the tree and refer to x as son of $f(x)$.

An algorithm for $s = t$

Step 1: For each vertex v_i compute the sum, R_i, of the r_j's in the subtree rooted at v_i.

Step 2: Let x be the current vertex and $x_1, \ldots, x_k$ be its sons. Assume, without loss of generality, that $R_1 \leqslant \cdots \leqslant R_k$. Starting with $x = s$ traverse the tree as follows:

- Go to the minimal indexed unvisited son of x, if one exists; else go back to $f(x)$.
- If $x = s$ and all sons of s were already visited, STOP.
- While traversing the tree, a delivery to any demand vertex x is made when the vehicle first visits x. A pickup from any supply vertex x is made when the vehicle last visits x, that is, when it leaves x to its father $f(x)$.

4.2. The tree case where s and t are predetermined

Let s and t be two distinct arbitrary vertices in the tree, let P denote the unique path between s and t in the tree, and let $\{x_1 = s, x_2, \ldots, x_k = t\}$ and $E(P)$ denote the vertex set and the edge set of the path P, respectively. Denote by $G_1, \ldots, G_k$ the collection of subtrees obtained upon deletion of $E(P)$ from the tree, where it is assumed that $x_i \in G_i$. A linear time algorithm for our vehicle routing problem on a tree is presented below.

An algorithm for $s \neq t$

Step 1: For each vertex $x_i \in P$ compute $r'(x_i)$ – the sum of the r_j's in G_i. This defines a request function r' on the vertices of P.

Step 2: Using the algorithm for the VRPD problem on a path, find an optimal tour T' for VRPD (P, r', N, s, t). Denote its length by $w(T')$. An optimal tour T for the tree is obtained from T' as follows. T coincides with T' on P. When T' performs a request by a vertex $x_i \in P$, T follows an optimal tour in the subtree G_i that starts and ends at x_i. (Observe that when T' calls for the vehicle to perform a request by $x_i \in P$, the vehicle must have enough capacity to fulfill all the requests generated by vertices in G_i.)

The length of the optimal tour, $w(T)$, generated by the algorithm is given by $w(T) := w(T') + 2\sum_{e_j \in E \setminus E(P)} l(e_j)$.

The correctness of the algorithm: Obviously, the tour introduced by the algorithm is feasible. Any feasible tour for the tree traverses every edge in the subtrees (i.e. in $E\backslash E(P)$) at least twice, and it induces a tour for VRPD (P,r',N,s,t) whose length is at least the length of an optimal tour for VRPD (P,r',N,s,t). Since the tour T generated by the algorithm traverses every edge in the subtrees exactly twice and the induced tour for VRPD (P,r',N,s,t) is optimal[2] we conclude that T is optimal. □

For simplicity, assume that the requirements at the leaves of the tree are not zero. In view of Lemma 5 and the above discussion of the correctness of the algorithm, we have:

Corollary 7. *Let T be an optimal tour in a tree graph with predetermined s and t. Then, every edge on the path between s and t is traversed either once or three times by T, and all other edges are traversed precisely twice.*

4.3. The case where t is endogenously determined

In order to find a vertex t for which the length of an optimal tour from a predetermined vertex s to t is minimal, one can apply, for all possible locations of t, the algorithm developed in Section 4.2 for predetermined s and t. This will result with a quadratic time algorithm. In this subsection we develop a linear time algorithm for the case where s is predetermined and t is endogenously determined.

Since s is predetermined, we root the tree in s, direct the edges from a father to its sons, and denote $e_i=(f(v_i),v_i)$, where $f(v_i)$ is the father of v_i in the tree.

An algorithm when t is endogenously determined

Step 1: For each vertex v_i compute the sum, R_i, of the r_j's in the subtree rooted in v_i. Denote by $\bar{R}_i$ the sum of the r_j's in the rest of the tree. Thus, $\bar{R}_i:=R_s-R_i$.

Step 2: For each edge e_i define $l'(e_i)$ as follows:

If $\bar{R}_i>c$ then $l'(e_i):=l(e_i)$: else, $l'(e_i):=-l(e_i)$.

Step 3: For each v_i compute $w(v_i)$, $w(v_i)=\sum(l'(e_i)$: e_i is on the s–v_i path). $w(s):=0$.

Step 4: $w(v_{i_0}):=\min_{i=1,\ldots,|V|}\{w(v_i)\}$.

Step 5: Find an optimal tour, T, for the tree from s to $t:=v_{i_0}$ using the algorithm developed in Section 4.2. Then, T is optimal for a tree graph with t endogenously determined and $w(T):=w(v_{i_0})+2\sum_{e_j\in E}l(e_j)$.

Implementation and correctness of the algorithm: Step 1 can be easily implemented in linear time by performing one pass on the tree from the leaves towards the root s. $R_i:=r_i+\sum_{j=1}^{k}R_{i_j}$ where $v_{i_1},\ldots,v_{i_k}$ are the sons of v_i. If v_i is a leaf, then $R_i:=r_i$. To implement Step 3 in linear time we perform a Breadth First Search starting from s. We set $w(s)$ to be zero, and for $v_i\neq s$ we set $w(v_i):=w(f(v_i))+w(e_i)$.

[2] Observe that if $r'(x_i)>0$ and $r(x_i)<0$, the optimal tour does not perform a delivery upon its first visit to the demand vertex x_i. The optimality of the tour follows from Observation 6.

Next, let us prove the correctness of the algorithm. Consider an arbitrary edge e_j, and assume that it is contained in the subpath $P_i = s - v_i$. Denote by $E(P_i)$ the edge set of P_i. Then, by Corollary 7, e_j is traversed by an optimal tour from s to v_i either once or three times. Consider the algorithm for predetermined s and t, developed in Section 4.2. It calls the algorithm on a path which computes the partial sum of the r_i's along the line. Note that the partial sum of the r_i's along the s–$f(v_j)$ path, used by the algorithm on a path, is equal to $\bar{R}_j$ computed in Step 1 above. Thus, if $\bar{R}_j \leqslant c$, e_j is traversed once. Otherwise, $\bar{R}_j > c$ and e_j is traversed three times. Thus, the length of an optimal tour, T_{v_i}, that starts at s and ends at v_i is

$$\begin{aligned} w(T_{v_i}) &= 2 \sum_{e_j \in E \setminus E(P_i)} l(e_j) + 3 \sum_{e_j \in E(P_i) \wedge (\bar{R}_j > c)} l(e_j) + \sum_{e_j \in E(P_i) \wedge (\bar{R}_j \leqslant c)} l(e_j) \\ &= 2 \sum_{e_j \in E} l(e_j) + \sum_{e_j \in E(P_i) \wedge (\bar{R}_j > c)} l(e_j) - \sum_{e_j \in E(P_i) \wedge (\bar{R}_j \leqslant c)} l(e_j) \\ &= 2 \sum_{e_j \in E} l(e_j) + w(v_i). \end{aligned}$$

The algorithm chooses t to be a vertex v_{i_0}, for which $\hat{w}(v_{i_0}) = \min_{i=1,\ldots,|V|}\{w(v_i)\}$. □

4.4. The tree case with s and t endogenously determined

In order to find a pair of vertices s and t, for which the length of an optimal tour from s to t is shorter than the length of an optimal tour from any vertex to any other vertex on the tree, one can apply, for all possible locations for s, the algorithm for a fixed s given in Section 4.3. This will result with a quadratic time algorithm. We develop below a linear time algorithm for the VRPD problem on a tree graph G for the case where both s and t are not predetermined.

First, we need to introduce the following notation. For an arbitrary edge $e = (u_1, u_2)$ in G, we denote by $H(u_1)$ and $H(u_2)$ the two components of the tree containing u_1 and u_2, respectively, which are obtained after the removal of e. Denote by $R(H(u_1))$ and $R(H(u_2))$ the total requirement of vertices contained in component $H(u_1)$ and $H(u_2)$, respectively.

An algorithm when s and t are endogenously determined

Step 1: Replace each edge, $e \in G$, $e = (u_1, u_2)$, in the tree graph G by two anti-parallel directed arcs $a_1 = (u_1, u_2)$ and $a_2 = (u_2, u_1)$, and denote by $\hat{G}$ the directed graph derived from G.

Step 2: For each directed arc $a = (u_1, u_2)$ in $\hat{G}$ derived from edge $e = (u_1, u_2)$ in G let

$$\ell'(a) = \ell'((u_1,\ u_2)) = \begin{cases} l(e) & \text{if } R(H(u_1)) > c, \\ -l(e) & \text{otherwise.} \end{cases}$$

Step 3: Choose an arbitrary vertex, v_0, as the root of the directed graph $\hat{G}$. A subgraph of $\hat{G}$ rooted at some vertex v is the directed graph that is associated with

the subtree of the tree G rooted at v, when G is also viewed as rooted at v_0. Now, in $\hat{G}$, rooted at v_0, find the length of a shortest path, P, and its end vertices s_P and t_P as follows:

(i) For every leaf vertex q, except the root, set $l(F(q)):=0$ and $l(T(q)):=0$.
(ii) Recursively, compute $l(F(v_i))$ – the length of a shortest path from v_i to a vertex contained in the subgraph rooted in v_i, $l(F(v_i)):=\min_{j=1,\ldots,k}\{l(F(v_{i_j}))+\ell'(v_i,v_{i_j})\}$, and $l(T(v_i))$ – the length of a shortest path from a vertex contained in the subgraph rooted in v_i to v_i, $l(T(v_i)):=\min_{j=1,\ldots,k}\{l(T(v_{i_j}))+\ell'(v_{i_j},v_i)\}$, where $v_{i_j},\ldots,v_{i_k}$ are the sons of v_i.
(iii) Compute $l(P(v_i))$ – the length of a shortest path which traverses v_i and is contained in the subgraph rooted at v_i. $l(P(v_i)):=l(F(v_i))+l(T(v_i))$ if both $l(F(v_i))$ and $l(T(v_i))$ are negative and $l(P(v_i)):=\min\{l(F(v_i)),l(T(v_i))\}$, otherwise.
(iv) $l(P):=\min_{i=1,\ldots,|V|}\{l(P(v_i))\}$. Set s_P and t_P to be the start and end vertices of the directed path P. An optimal tour for the tree from s_P to t_P, generated by the algorithm developed in Section 4.2, is an optimal tour for a tree when both s and t are endogenously determined.

Implementation and correctness of the algorithm: Step 3(ii) is implemented in linear time, by performing one pass on $\hat{G}$ from the leaves backwards to the root v_0. By Section 4.2, Step 3(iv) is also carried out in linear time. Thus, the above algorithm is linear.

The arc weights introduced in Step 2 and the fact that we have to find a shortest directed path in $\hat{G}$ are justified by a proof similar to the correctness proof given in Section 4.3. The choice of the root, v_0, for $\hat{G}$ in Step 3 is arbitrary, and is done just to enable us to carry out the computation in linear time. In Step 3(ii) we find recursively, for each vertex v, the length of a shortest directed path which traverses v and is contained in the subgraph of $\hat{G}$ rooted at v. The correctness proof follows since the shortest path we seek traverses some vertex v of G and is contained in the subgraph of $\hat{G}$ rooted therein. □

5. The VRPD problem on a cycle graph

In this section we consider the VRPD problem on a cycle graph C, denoted VRPD(C,r,N). In Section 5.1, we develop algorithms for the VRPD(C,r,N) problem in the case where both the starting vertex s and the terminal vertex t of the tour are endogenously determined by the algorithm, for both cases $s=t$ and $s\neq t$. In Section 5.2, we develop an algorithm which finds an optimal tour for the case where the locations of both s and t are predetermined, VRPD(C,r,N,s,t). We start with some notation and lemmas.

Throughout this section we assume that the vertices of the cycle graph $C, v_1,\ldots,v_{|V|}$, are ordered clockwise, $e_i=(v_i,v_{i+1})\in E$, $1\leqslant i<|V|$, and $e_{|V|}=(v_{|V|},v_1)$. We denote the length of the cycle by $l(C)=\sum_{i=1}^{|V|}l(e_i)$ and the length of a feasible tour, T, by $w(T)$, which is the sum of lengths of all the edges traversed by T. We assume,

without loss of generality, that there exists at least one supply vertex in C, otherwise the problem reduces to the travelling salesman problem on a cycle graph, which is trivial. We assume, without loss of generality, that $r(v_i) \neq 0$, $1 \leqslant i < |V|$. Otherwise, we can delete any vertex v_i for which $r(v_i) = 0$ along with its two incident edges, and connect its two adjacent vertices v_{i-1} and v_{i+1} by a new edge whose length equals the sum of the lengths of the two edges deleted.

Since any feasible tour in C must visit all the vertices we have:

Observation 8. *At most one edge in C is not traversed by a feasible tour for a $VRPD(C,r,N)$ problem.*

Let $\{S_i\}$ denote the sequence of partial sums of the sequence $\{r_j\colon 1 \leqslant j \leqslant |V|\}$. Thus, $S_i = \sum_{j=1}^{i} r_j$. Let $\hat{v}$ be a vertex for which $\max_{i=1,\ldots,|V|}\{S_i\}$ is attained. Then, it is easy to see that the following result holds.

Lemma 9. *A feasible tour for a $VRPD(C,r,N)$ problem, which starts and ends at $\hat{v}$ traverses all the edges in C exactly once.*

A proof of Lemma 9, not using the sequence of partial sums, was given by Mosheiov [8].

5.1. The $VRPD(C,r,N)$ problem with s and t endogenously determined

An algorithm for a cycle graph with s and t endogenously determined

Step 1: For every edge $e_i \in C$, for which either there exists a feasible tour that starts at v_{i+1} and goes clockwise to v_i, or starts at v_i and goes counterclockwise to v_{i+1} traversing every edge exactly once, let $w(T_{e_i}) := l(C) - l(e_i)$. Let $w(T_{k_0}) := \min_i\{w(T_{e_i})\}$.

Step 2: For the rest of the edges e_j in C compute $w(T_{e_j})$, the length of an optimal tour, T_{e_j}, that does not traverse e_j. Observe that the removal of an edge e_j induces a VRPD problem on the path $C \setminus e_j$, in which s and t are endogenously determined. Thus, T_{e_j} and $w(T_{e_j})$ can be generated by the algorithm developed in Section 4.4. Let $w(T_{j_0}) := \min_{j=1,\ldots,|V|}\{w(T_{e_j})\}$.

The length of the optimal tour, $w(T)$, generated by the algorithm is given by $w(T) := \min\{w(T_{j_0}), w(T_{k_0})\}$. That is, if $w(T_{j_0}) < w(T_{k_0})$ then an optimal tour is $T := T_{j_0}$, else $T := T_{k_0}$.

Implementation and correctness of the algorithm: Step 1 can be implemented in linear time, but Step 2 involves $O(|V|)$ calls (in the worst case) of the linear time algorithm developed in Section 4.4. Thus, the time complexity of the above algorithm is $O(|V|^2)$.

Steps 1 and 2 check all the feasible tours that traverse all edges in C except one. Indeed, in Step 2 the algorithm generates for each j, an optimal tour for a VRPD problem on a path $C \setminus e_j$, where $s \neq t$ are endogenously determined. On the other

hand, Lemma 9 guarantees the existence of a feasible tour that traverses every edge in C exactly once for some $s = t = v_j$. Thus, there are feasible tours that traverse all the edges except one, exactly once. Any such tour starts at a vertex v_i for which $\max S_i$ is attained (see Lemma 9), and ends at v_{i-1}, if the corresponding tour travels clockwise, or at v_{i+1} if the tour travels counterclockwise. Such tours are examined by the algorithm in Step 1. Note that one could have applied Step 2 to all the edges of C. However, it is more efficient to find optimal tours which do not traverse edges satisfying the condition in Step 1. Indeed, such tours can be found in constant time, instead of linear time per edge if Step 2 is applied. □

Observation 8 and Lemma 9 suggest the following linear time algorithm for a VRPD(C, r, N) problem with $s = t$ endogenously determined.

An algorithm for a cycle graph with $s = t$

Step 1: For every edge $e_j \in C$ let $w(T_{e_j}) := 2(l(C) - l(e_j))$. Observe that $w(T_{e_j})$ is the length of a shortest tour for the VRPD problem defined on the path derived from C after the elimination of e_j, with $s = t = v_j$. Let $w(T_{j_0}) := \min_{j=1,\ldots,|V|}\{w(T_{e_j})\}$.

Step 2: Find an index k_0, such that a feasible tour that starts and ends in v_{k_0} and travels clockwise, traverses every edge exactly once, with $s = t = v_{k_0}$. Denote this tour by T_{k_0}, $w(T_{k_0}) = l(C)$.

Let $w(T) = \min\{w(T_{j_0}), w(T_{k_0})\}$. If $w(T_{j_0}) < w(T_{k_0})$, then an optimal tour T starts at v_{j_0+1}, travels to v_{j_0} clockwise performing all delivery requests and then returns counterclockwise from v_{j_0} to v_{j_0+1} performing all pickup requests. Else, $T := T_{k_0}$ is an optimal tour.

Implementation and correctness of the algorithm: Clearly, the above algorithm can be carried out in linear time. To verify its correctness, observe that by Observation 8, at most one edge in C is not traversed by a feasible tour on C. The removal of an edge e_j from C induces a VRPD problem on a path $C \setminus e_j$, in which $s = t$ are endogenously determined. As observed in Section 4.1, the length of an optimal tour on a tree with $s = t$ is independent of the choice of s, and is always equal to twice the length of the tree. Thus, for a VRPD problem on $C \setminus e_j$, we can choose $s = t$ to coincide with v_{j+1}.

It remains to check all the feasible tours that traverse all edges in C at least once. Lemma 9 guarantees the existence of a feasible tour that traverses every edge in C exactly once, and all such tours, whose length is $l(C)$, are found in Step 2 of the algorithm. □

5.2. The VRPD problem on a cycle graph with predetermined s and t

Before analyzing the case where s and t are predetermined, we construct below a parametric algorithm for the VRPD problem on a path, which computes the length

of optimal tours for all nonnegative initial capacities c in $O(|V|\log|V|)$ time. This parametric algorithm is used in our $O(|V|^2\log|V|)$ algorithm for the VRPD problem on a cycle when s and t, $s \neq t$, are predetermined, which is described in Section 5.2.1 below.

An algorithm for a path with $s \neq t$ and parametric initial capacity

Step 1: Calculate the partial sum S_j for each vertex v_j of the path, and sort these sums, to obtain the subsequence, $S'_1, S'_2, \ldots, S'_k$, of all distinct values of the S_j's arranged in an ascending order. For each S'_i, let $E(S'_i) = \{e_j = (v_j, v_{j+1}) : S_j = S'_j\}$.

Step 2:

- If the initial available capacity of the vehicle $c < S_t$, where S_t is the partial sum at t, the tour is infeasible.
- If the initial available capacity of the vehicle $c \geqslant S'_k$, an optimal tour T_k will traverse the whole path exactly once, and have the length $w(T_k) = d(s,t)$, where $d(s,t)$ is the length of the path from s to t.
- For $i = k-1, \ldots, 1$ and an initial available capacity of the vehicle $c \in [S'_i, S'_{i+1})$, an optimal tour T_i traverses three times all segments traversed three times by T_{i+1} and all edges in $E(S'_i)$. The length of the tour is $w(T_i) = w(T_{i+1}) + 2\sum_{e\in E(S'_i)} l(e)$.

Since the partial sums and the sets $E(S'_i)$ can be found in linear time, the complexity of the algorithm is the complexity of sorting the partial sums, which can be done in $O(|V|\log|V|)$ time. The correctness of the algorithm follows from Lemma 5 and the correctness proof of the algorithm for a path with a fixed initial capacity which demonstrated that an edge (v_i, v_{i+1}) for which $S_i \leqslant c$ is traversed once, and it is traversed three times if $S_i > c$. □

Lemma 10. *For a path L, denote by L' the graph in which the supplies $r(v_{i_1}), \ldots, r(v_{i_k})$ at the first k supply vertices in the direction s–t are replaced with* 0. *An optimal tour T' on the path L' with an initial vehicle capacity c coincides with an optimal tour T on the path L with an initial vehicle capacity $c + \sum_{j=1}^{k} r(v_{i_j})$.*

Proof. If the sum of all deliveries prior to v_{i_k+1} is S ($S \leqslant 0$), T' will traverse all edges prior to v_{i_k+1} only once, and the vehicle capacity upon reaching v_{i_k+1} will be $c - S$.

T will also traverse all edges prior to v_{i_k+1} only once. Further, the net load picked up by the vehicle until reaching v_{i_k+1} is $\sum_{j=1}^{k} r(v_{i_j}) - S$. Therefore, upon reaching v_{i_k+1}, the vehicle's capacity is also equal to $c - S$. Since the remaining parts of L and L' are identical, T and T' will also coincide on the remaining part of the path. □

5.2.1. The VRPD problem on a cycle graph when s and t are predetermined and $s \neq t$

For a cycle C with predetermined $s \neq t$, let A and B denote the two s–t paths contained in the cycle.

Proposition 11. *Each feasible tour, T, on a cycle that visits all vertices has to have one of the following forms*:

Form (i): *One edge in one of the s–t paths is not traversed at all by T and all other edges in that s–t path are traversed at least twice*,

Form (ii): *Every edge in C is traversed at least once, and every edge in one of the s–t paths is traversed at least twice*.

Proof. Let vertices v_i, v_{i+1} be on A, $e=(v_i,v_{i+1})$:

(i) Any tour that does not traverse e but visits all vertices in C must contain a subtour wherein it travels from s along A to v_i, returns back to s, goes along B to t, continues along A to v_{i+1}, and then returns along A to t. All edges on A except e are traversed at least twice and e is not traversed.

(ii) Suppose that e is traversed by a tour exactly once, and that each other edge is traversed at least once. If e is traversed in the direction from s to t, the tour is going along A in direction from s to t, and does not return to s along A once it reached t. In order for it to visit all vertices along B, the tour must traverse all edges in a subpath (possible empty) of B at least twice before entering A (goes along B in the direction from s to t, until some vertex u_1, then back to s and continues along A), and all edges in a subpath (possible empty) of B at least twice after leaving A (goes along B in the direction from t to s, until some vertex u_2, then back to t). If e is traversed in the direction from t to s, all edges in B are traversed at least once before the tour first reached t. Since e is traversed exactly once, the tour does not return to t along A. So all edges in B are traversed at least once more on the way from s to t. Either way, all edges in B are traversed at least twice. □

Since every tour must have one of the forms mentioned above, so does an optimal tour.

If an optimal tour traverses all edges at least once, it can be done in four ways, as we show below. Denote by A the s–t path in which every edge is traversed by an optimal tour at least twice. Let T be a feasible tour and let Q be a subpath of T from v_1 to v_2. If Q is contained in A (resp. B) and $v_1,v_2\in\{s,t\}$, then Q will be referred to as a *visit* by T in A (resp. B), and if $v_1\neq v_2$, then Q will be referred to as a *complete visit* by T in A (resp. B).

Theorem 12. *The subpath A can be traversed by an optimal tour, T, in four possible forms*:

Form (1): *There are precisely two complete visits in A, both of which start at s*.

Form (2): *There are two complete visits in A. The first complete visit starts at s, the second complete visit starts at t, and there may be some other incomplete visits in A after the complete visits therein*.

Form (3): *There are two complete visits in A. The first complete visit starts at t, the second complete visit starts at s, and there may be another incomplete visit in A which precedes the two complete visits therein*.

Form (4): *There is one visit in A, which enters at s and leaves at s, and there is another visit in A which enters at t and leaves at t.*

In order to prove Theorem 12, we need the following lemmas:

Lemma 13. *If a tour traverses all edges in one of the s–t paths at least twice, and all edges in the other s–t path at least three times, the tour cannot be optimal.*

Proof. Without loss of generality, let us assume that all edges in A are traversed at least three times by a tour T, and let $v \in A$ be the supply vertex on A closest to s. The tour that starts along B from s to t and continues along A from t to s doing all deliveries, continues along B from s to t performing pickups, along A from t to v idly, and then back to t along A doing pickups, is a feasible tour shorter than T, so T cannot be optimal. □

Lemma 14. *If an optimal tour, T, starts along A and leaves A for the first time at s, its next visit in A must start at t.*

Proof. Suppose T is an optimal tour which starts along A and leaves A for the first time at s, but, on the contrary, its next visit in A starts at s. Then, T enters B at s after leaving A, and must leave B at s as well. Let $A_1 = (s, u_i)$ denote the segment of A visited by T during its first visit in A, and $B_1 = (s, v_j)$ the segment of B visited by T during its first visit in B:

- If T made only deliveries on B_1, then it did not need to enter A before visiting B. Indeed, it could start by entering B at s, doing only deliveries along B_1, exit B at s and combine its two visits in A into one visit. Such a tour will be shorter than T.
- If some pickups were performed on B_1, it follows from Assumption 2 that the tour will not visit again the vertices where pickups were performed. If v_j was a pickup vertex, this would mean that the edge (v_j, v_{j+1}) would remain untraversed. That contradicts the assumption that each edge is traversed at least once, so v_j must be a delivery vertex. Let us denote by v_k the supply vertex on B_1 closest to v_j. Using the same argument, if a pickup at v_k is performed during the first visit in B_1, edge (v_j, v_{j+1}) will not be traversed by T. Let v_l be the supply vertex closest to s on B_1 where pickup was not performed during the first visit of T in B (possible $v_l = v_k$); T did not perform any pickup at supply vertices between v_l and v_k. To perform the pickup at v_l, T must enter B at t and leave it at t. That means that T must go along A from s to t, then continue along B from t until v_l and return to t along B. Since all edges in A must be traversed at least twice, T must then go along A towards s, past u_i to some supply vertex u_h, and then return back to t. However, this implies that T traverses all edges on A at least three times and all edges on B at least twice. It follows from Lemma 13 that T cannot be optimal. □

Lemma 15. *If the first visit in A by an optimal tour, T, starts and ends at s, and the second visit in A starts and ends at t, T will not visit A again.*

Proof. The whole length of B is traversed at least once before T enters A for its second visit. Therefore, all deliveries along B were performed before the beginning of T's second visit in A. Any visit by T in B, which starts and ends at t, done after T's second visit in A, will increase the load of the vehicle. Thus, such a visit, if exists, can be delayed until the end of the tour. Therefore, if there is a third visit by T in A, the starting vertex of such a visit must be s. If the third visit in A is not complete, T must have traversed all edges in B at least three times and, by assumption, all edges in A have been traversed twice. If the third visit in A is a complete visit, T must have traversed all edges in A at least three times (by assumption each edge therein must be traversed at least twice), and each edge in B at least twice. So, by Lemma 13, such a tour cannot be optimal. Therefore, T will not contain a third visit in A. □

Lemma 16. *If the first visit in A by a tour, T, starts and exits at t, and the second visit in A starts and exits at s, the tour T is not optimal.*

Proof. The whole length of B is traversed at least twice before T enters A for its second visit. After the second visit in A, we have two possible cases:

- All edges in A were traversed at least twice. Then, from s, T can either go to t along B, or, if no pickups were yet performed in A, T can go from s to t along A. In both cases, non optimality of T follows from Lemma 13.
- Edges between some vertices u and v in A were not traversed. However, by assumption all edges in A must be traversed at least twice. Thus, T must have at least one more visit in A. However, one can easily verify that these additional visits in A will imply that either T will have to traverse each edge in B at least three times and each edge in A at least twice, or T will have to traverse each edge in A at least three times and each edge in B at least twice. In both cases, non optimality of such a tour follows from Lemma 13. □

Lemma 17. *A tour T that has two successive incomplete visits in A (or B) starting and ending in the same vertex, either s or t, cannot be optimal.*

Proof. We will prove the result for the s–t path A, and a starting and ending vertex s. All pickups and deliveries that were performed during the tour's two visits in A could have been done during just one visit. Indeed, let v_1 and v_2, respectively, denote the furthest points from s on A reached by T during its first and second incomplete visit in A, and let v_3 denote the furthest point from s on B reached by T during its incomplete visit in B done in between its two incomplete visits in A. Then, the total length of those incomplete visits by T is equal to $2(d(s,v_1)+d(s,v_2)+d(s,v_3))$, where $d(s,v_1)$ and $d(s,v_2)$ are calculated along A and $d(s,v_3)$ is calculated along B. Now, if the net load picked up by T during its incomplete visit in B is negative (resp. positive), a shorter

tour can be obtained by combining the two incomplete visits in A into one incomplete visit, which is done after (resp. before) the incomplete visit in B. The length of that part of the tour will be $2(\max\{d(s,v_1),d(s,v_2)\}+d(s,v_3)) < 2(d(s,v_1)+d(s,v_2)+d(s,v_3))$. □

Lemma 18. *A complete visit in A by an optimal tour T must be followed by another complete visit in A.*

Proof. After a complete visit of T in A, all deliveries along A were performed and only pickups (if any) are left for subsequent visits. On the other hand, it follows from Lemmas 15–17 that T contains at most one incomplete visit, Q_1, in A, which was carried out before the first complete visit therein. However, pickups were not done in any such incomplete visit in A. Indeed, by Assumption 2, pickups can be delayed to the last visit to a pickup vertex. Thus, at the end of the complete visit in A, there are edges in A that were traversed only once. Since we assume that every edge in A is traversed at least twice, T must contain at least one other visit, Q, in A:

- If, at the outset, A did not contain supply vertices, then, since T is assumed to be an optimal tour and there are no deliveries left to be performed in A, Q must be a complete visit in A. Indeed, if T contains an incomplete visit in A in which neither deliveries or pickups are performed, then T is not optimal.
- If some pickups are left in A after the complete visit therein, and Q is an incomplete visit which starts and ends at the same vertex, say s, and performs its last pickup at v_i, it did not visit any vertex after v_i. Since some pickups were performed in Q, all subsequent visits in A cannot be complete and the tour must go to t along B. If some pickup is still left in A after Q, the last incomplete visit in A, Q_1 must start and end at t, and it cannot go beyond v_{i+1}, the vertex following v_i on A in the direction from s to t. If Q_1 did not exist, or Q_1 started at s and ended at some vertex v_j that is closer to s on A than v_i, or Q_1 started at t and ended at some vertex v_k that is closer to t on A than v_{i+1}, then (v_i, v_{i+1}) will be traversed only once, which contradicts our assumption that all edges in A are traversed at least twice. Otherwise, all edges in A will be traversed at least three times, and all edges in B at least twice, and non optimality of T in that case follows from Lemma 13. Thus, Q must be a complete visit. A similar conclusion is reached when Q is an incomplete visit which starts and ends at t, and the proof follows. □

Lemma 19. *A tour that makes two complete visits in A which start at t cannot be optimal.*

Proof. After the first complete visit in A which started at t, all edges in both A and B were traversed at least once, and all deliveries in C were performed. A tour which contains a second complete visit in A which starts at t will traverse all edges on one s–t path of C at least three times, and all edges on the other s–t path at least twice. Therefore, by Lemma 13, such a tour cannot be optimal. □

Proof of Theorem 12. An optimal tour, T, can start either along A or along B. If it starts along A, then A is entered for the first time at s:

(I) If T, on its first visit in A, leaves A at s, we know from Lemma 14 that the second visit must start at t. It can leave A, in the second visit, either at s or at t:
- If the second visit exits at t, it follows from Lemma 15 that T has Form (4).
- If the second visit exist at s, it follows from Lemma 18 that there is a third visit to A which is complete:
  - If the third visit starts at s and ends at t, T has Form (3). All deliveries on A and B were carried out before the beginning of the third visit in A, so all pickups are performed during the third visit in A and A will not be visited again.
  - If the third visit starts at t and ends at s, by Lemma 19, T cannot be optimal.

(II) If an optimal tour, T, on its first visit in A, exits A at t, all deliveries in A were carried out and only pickups are left in A. The second visit in A can start either at s or at t:
- If the second visit starts at s, it must end at t because of Lemma 18, and T has Form (1). All deliveries in A and B were done before the tour's second visit in A, so all the pickups are carried out during the tour's second visit in A, and A will not be visited again.
- If the second visit starts at t, it must end at s because of Lemma 18, and T has Form (2). There could be other visits in A, which are not complete, if some pickups were left in A after the second visit therein.

If the tour, T starts along B, A can be entered for the first time either at s or at t.

(III) If A is entered at s, T has left B at s:
- If T exits A at s, it is entering B again:
  - If T has not performed any pickups during its first visits in B or A, it is not optimal since all deliveries in B done during the tour's first visit therein could be done during its second visit in B.
  - If some pickups were performed during T's first visit in A, T cannot have subsequent complete visits in A, since a pickup from a supply vertex is left for the last visit by the tour to that vertex. Moreover, its second visit in A must start and end at t, and by Lemma 15, there are no other visits in A. At the end of the tour's first visit in A, B is traversed from s to t in an optimal manner. However, since the tour started along B, it follows that there was not enough available capacity on the vehicle to perform all the pickups during the first visit by T in A. That is, the sum of all pickups and deliveries performed during the first visit by T in A is strictly positive. Thus, the length of an optimal subtour in B from s to t, when the vehicle starts with its initial capacity and performs all requests in B is shorter than the length of that part of T used to traverse edges in B. This suggests that T is not optimal, and a shorter tour can be obtained by a vehicle which starts

at s and traverses B in an optimal manner, while performing all requests therein. Then, upon reaching t, the vehicle enters A and travels along A from t to the vertex closest to s in A doing deliveries, and then it returns to t along A while performing only pickups.

– If some pickups were performed during the first visit in B, T cannot have subsequent complete visits in B, since a pickup from a supply vertex is left for the tour's last visit in that vertex. So, since the tour must eventually reach t, no pickups were performed during its first visit in A, and the tour must exit B at s after its second visit therein. However, by Lemma 17, such a tour is not optimal.

- If T exits its first visit in A at t, all deliveries on A were done and only pickups therein remain to be performed. The second visit in A can start at s or at t:
  – If the second visit starts at s, it must end at t because of Lemma 18, and T has Form (1). All deliveries on A and B were done before the second visit in A, so all the pickups are performed during the second visit in A, and A will not be entered again.
  – If the second visit starts at t, then it follows from Lemma 18 that it must end at s. Therefrom, the tour traverses B in the direction from s to t. Thus, the vehicle did not pickup supply from any vertex in B in its first incomplete visit therein. Now, since the tour started with an incomplete visit in B, starting and ending at s, the net supply picked up by the vehicle from its first entry into A at s on its first visit, until its exit at s at the end of its second visit in A, must be positive. Thus, the length of an optimal subtour in B from s to t, when the vehicle starts with its initial capacity and performs all requests in B is shorter than the length of that part of T used to traverse edges in B. This suggest that T is not optimal, and a shorter tour can be obtained by a vehicle which starts at s and traverses B in an optimal manner, while performing all requests therein, and then the tour performs one incomplete visit in A, starting and ending at t, while reaching until the closest vertex to s on A.

(IV) If A is entered at t, T has traversed all the edges in B at least once and has exited B at t:

- If T exits A at s, the second visit can start at s or at t:
  – If the second visit starts at s, it must end at t because of Lemma 18, and T has Form (3). All deliveries on A and B were done before the second visit in A, so all the pickups are performed during the second visit in A, and A will not be entered again.
  – If the second visit starts at t, it must end at s because of Lemma 18. It follows from Lemma 19 that T cannot be optimal.
- If T exits A at t, its second visit in A can start as s or at t:
  – If the second visit starts at s, all the edges in B were traversed at least twice before the beginning of the second visit in A. Since we assume that all edges in A are traversed at least twice, T will eventually traverse all edges in one

of the s–t paths at least twice, and all edges in the other s–t path at least three times. It follows from Lemma 13 that T cannot be optimal.
If, on the other hand, all edges in B are traversed at least twice, and T traverses some of the edges in A only once, we have obtained an analogue version of the second case in (II) above, where B is replaced with A. Since the first complete visit in B has started at s, and the second has started at t, T has Form (2).

– If the second visit starts at t, then either there was an incomplete visit in B, starting and ending at t, between the first and the second visits in A, or there was no such visit in B. In the former case, since all deliveries in B were performed in the first (complete) visit therein, such an incomplete visit can only reduce the available capacity of the vehicle, and thus can be delayed until the end of the tour. In the later case, the first visit in A can be combined with the second (complete) visit in A. In both cases we conclude that if the second visit starts at t, the tour is not optimal. □

To find an optimal tour, we must check all possible tours that have Form (i) in Proposition 11, or Forms (1)–(4) in Theorem 12, excluding the cases that were shown to be non-optimal in the proof of Theorem 12.

An algorithm for a cycle graph with $s \neq t$

Step 1 (In this step we calculate, in $O(|V|^2)$ time, the length of a shortest tour which is of the Form (i) in Proposition 11.):

- For every edge (v,w) in C, use the algorithm for a tree to find an optimal tour $T^1_{v,w}$ on the v–w path derived from C by eliminating (v,w).
- Let $w(T_1) := \min\{w(T^1_{v,w})\colon (v,w) \text{ in } C \text{ and } v,w \notin \{s,t\}\}$.

Step 2 (In this step we calculate, in $O(|V|^2 \log |V|)$ time, the length of a shortest tour which is of Form (1) in Theorem 12. The general description of this tour is presented in Fig. 1):

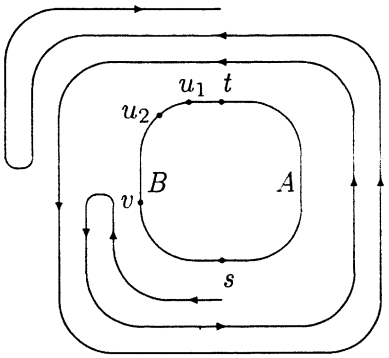

Fig. 1. Form (1).

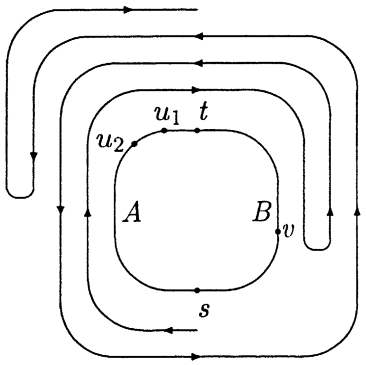

Fig. 2. Form (2).

- For every demand vertex $v \neq t$ denote by B (resp. A) the s–t path that contains[3] (resp. does not contain) v. Let S_v denote the absolute value of the total demand in all demand vertices contained in A and in the subpath of B between s and v. Let $u_1, u_2, \ldots, u_k$ denote the supply vertices in B, arranged in an increasing distance from t, $u_1 \neq t$, and let $S_{vu_j} = S_{vu_{j-1}} + r(u_j)$, $j = 1, \ldots, k$, $s_{vu_0} = S_v$. Consider the VRPD problem defined on a path L which coincides with B, but in which the demand in all demand vertices contained in the s–v subpath of B were set to zero, and L is traversed in the direction from t to s:
  (a) Use the parametric algorithm for the VRPD problem on a path, with parametric available capacity α, to determine the length of a shortest tour, $T^2_{v,\alpha}$, on L, for $\alpha = c + S_{vu_0}, c + S_{vu_1}, \ldots, c + S_{vu_k}$, where c is the initial available capacity when the vehicle starts at s.
  (b) Let $T^2_v = \arg\min\{w(T^2_{v,c+S_{vu_j}}) + 2d(u_j, t)\colon j = 0, \ldots, k\}$, where $d(u_j, t)$ is the distance between u_j and t on B and $d(u_0, t) = 0$.
  (c) Let $w(T^2) := \min_v [w(T^2_v) + 2d(s, v) + 2l(A)]$, where $d(s, v)$ is the distance between s and v on B.

Step 3 (In this step we calculate, in $O(|V|^2 \log |V|)$ time, the length of a shortest tour which is of Form (2) in Theorem 12. The general description of this tour is presented in Fig. 2):

- For every demand vertex $v \neq s$, denote by B (resp. A) the s–t path that contains[4] (resp. does not contain) v. Let S_v denote the absolute value of the total demand in all demand vertices contained in A and in the subpath of B between v and t. Let $u_1, u_2, \ldots, u_k$ denote the supply vertices in A, arranged in an increasing distance from

[3] For $v = s$, this part of the algorithm is performed twice. Once when the tour T is as shown in Fig. 1, in which the initial incomplete visit in B is eliminated. The other case is obtained when the roles of A and B are reversed.

[4] For $v = t$, this part of the algorithm is performed twice. Once when the tour T is as shown in Fig. 2, in which the first incomplete visit in B, starting and eding at t, is eliminated. The other case is obtained when the roles of A and B are reversed.

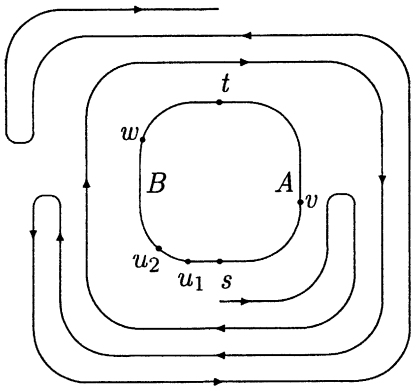

Fig. 3. Form (3).

t, $u_1 \neq t$, and let $S_{vu_j} = S_{vu_{j-1}} + r(u_j)$, $j = 1,\ldots,k$, $S_{vu_0} = S_v$. Consider the VRPD problem defined on a path L, which coincides with the circle disconnected at t. Thus, one end point of L is t, the other is t', and the $(t$–$t')$ path consists of subpath A concatenated with subpath B. However, the demand in all demand vertices contained in A and in the t–v subpath of B are set to zero, while the demand/supply of all other vertices in L remains equal to their values in C.

(a) Use the parametric algorithm for the VRPD problem on a path, with parametric available capacity α, to determine the length of a shortest tour, $T^3_{v,\alpha}$, on L, for $\alpha = c + S_{vu_0}, c + S_{vu_1}, \ldots, c + S_{vu_k}$, where c is the initial available capacity when the vehicle starts at s.

(b) Let $T^3_v = \arg\min\{w(T^3_{v,c+S_{vu_j}}) + 2d(u_j,t)\colon j=0,\ldots,k\}$, where $d(u_j,t)$ is the distance between u_j and t on A and $d(u_0,t) = 0$.

(c) Let $w(T^3) := \min_v[w(T^3_v) + 2d(v,t) + l(A)]$, where $d(v,t)$ is the distance between v and t on B.

Step 4 (In this step we calculate, in $O(|V|^2 \log |V|)$ time, the length of a shortest tour which is of Form (3) in Theorem 12. The general description of this tour is presented in Fig. 3):

- For every demand vertex $v \neq t$, denote by A (resp. B) the s–t path that contains[5] (resp. does not contain) v. Let S_v denote the absolute value of the total demand in all demand vertices contained in the subpath of A between s and v. Let $u_1, u_2, \ldots, u_k$ denote the supply vertices in B, arranged in an increasing distance from s, $u_1 \neq s$, and let $S_{vu_j} = S_{vu_{j-1}} + r(u_j)$, $j = 1,\ldots,k$, $S_{vu_0} = S_v$.

[5] For $v = s$, this part of the algorithm is performed two times. Once when the tour T is as shown in Fig. 3, in which the initial incomplete visit in A is eliminated. The other case is obtained when the roles of A and B are reversed.

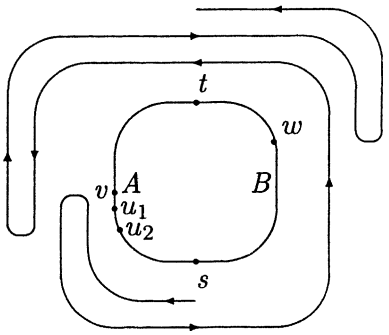

Fig. 4. Form (4).

We observe that for some pairs (v,u_j), there is no feasible solution for a VRPD problem on B, from s to t, with initial capacity $\alpha = c + S_{vu_j}$, in which the demand at all demand vertices in the subpath (s,u_j) of B was set to zero. That is, the net supply in the subpath (u_j,t) of B, excluding the supply at u_j, exceeds α. In these cases, there exists a supply vertex w on B such that according to an optimal tour, displayed in Fig. 3, the vehicle will not pick up the supply from all supply vertices on the subpath (w,t) during this part of the tour. Rather, the vehicle will leave this supply behind, will enter A at t for a complete visit therein and will follow the route shown in Fig. 3. Upon the return of the vehicle to t, it will enter B at t for an incomplete visit in order to pickup up the supply left behind in the subpath (w,t). Therefore, in (a) below, when we apply the parametric algorithm for the VRPD problem defined on B, we increase the *demand* at t to a large enough value M, say M is equal to the total supply on B. The justification for such a modification follows from the algorithm on the tree for $s \neq t$:

(a) Use the parametric algorithm for the VRPD problem on a path, with parametric available capacity α, to determine the length of a shortest tour, $T^4_{v,\alpha}$, on B, for $\alpha = c + S_{vu_0}, c + S_{vu_1}, \ldots, c + S_{vu_k}$, where c is the initial available capacity when the vehicle starts at s, and $r(t) = -(\sum r(v_j)\colon v_j \in B \text{ and } r(v_j) > 0)$.

(b) Let $T^4_v = \arg\min\{w(T^4_{v,c+S_{vu_j}}) + 2d(s,u_j)\colon j = 0,\ldots,k\}$, where $d(s,u_j)$ is the distance between s and u_j on B and $d(s,u_0) = 0$.

(c) Let $w(T^4) := \min_v[w(T^4_v) + 2d(s,v) + 2l(A)]$, where $d(s,v)$ is the distance between s and v on A.

Step 5 (In this step we calculate, in $O(|V|^2 \log |V|)$ time, the length of a shortest tour which is of Form (4) in Theorem 12. The general description of this tour is presented in Fig. 4):

- For every demand vertex $v \notin \{s,t\}$ for which there exists at least one supply vertex between s and v, denote by A (resp. B) the s–t path that contains (resp. does not contain) v. Let S_v denote the absolute value of the total demands minus total supplies in all vertices contained in the subpath of A between s and v. Let $u_1, u_2, \ldots, u_k$ denote the supply vertices in the subpath of A between s and v, arranged in an

increasing distance from v, and let $S_{vu_j} = S_{vu_{j-1}} + r(u_j)$, $j = 1,\ldots,k$, $S_{vu_0} = S_v$. Let $i = \min\{j = 1,\ldots,k: S_{vu_j} \geqslant 0\}$.

Similar to the observation made in Step 4, for some pairs (v,u_j), there is no feasible solution for a VRPD problem on B, from s to t, with initial capacity $\alpha = c + S_{vu_j}$, in which the demand at all demand vertices in the subpath (s,u_j) of B was set to zero. That is, the net supply in the subpath (u_j,t) of B, excluding the supply at u_j, exceeds α. In these cases, there exists a supply vertex w on B such that according to an optimal tour, displayed in Fig. 4, the vehicle will not pick up the supply from all supply vertices on the subpath (w,t) during this part of the tour. Rather, the vehicle will leave this supply behind, will enter A at t for an incomplete visit therein and will follow the route shown in Fig. 4. Upon the return of the vehicle to t, it will enter B at t for an incomplete visit in order to pickup up the supply left behind in the subpath (w,t). Therefore, in (a) below, when we apply the parametric algorithm for the VRPD problem defined on B, we increase the *demand* at t to a large enough value M, say M is equal to the total supply on B. The justification for such a modification follows from the algorithm on a tree for $s \neq t$:

(a) Use the parametric algorithm for the VRPD problem on a path, with parametric available capacity α, to determine the length of a shortest tour, $T^5_{v,\alpha}$, on B, for $\alpha = c + S_{vu_i}, c + S_{vu_{i+1}},\ldots, c + S_{vu_k}$, where c is the initial available capacity when the vehicle starts at s, and $r(t) = -(\sum r(v_j)$: $v_j \in B$ and $r(v_j) > 0)$.

(b) Let $T^5_v = \arg\min\{w(T^5_{v,c+S_{vu_j}}) + 2d(u_j,v)$: $j = 1,\ldots,k\}$, where $d(u_j,v)$ is the distance between u_j and v on A.

(c) Let $w(T^5) := \min_v[w(T^5_v) + 2l(A)]$.

The optimal tour, T, generated by the above algorithm is $T = \arg\min_i\{w(T^i)$, $i = 1,2,3,4,5\}$.

Implementation and correctness of the algorithm: Each step involves $O(|V|)$ calls of an algorithm of complexity $|V|\log|V|$, so the total time complexity of the above algorithm is $O(|V|^2\log|V|)$.

The correctness of the algorithm follows from the proofs of Proposition 11 and Theorem 12. Indeed, an optimal tour is either of the form (i) in Proposition 11 or its structure coincides with one of the tours displayed in Figs. 1–4, as proven in Theorem 12.

Step 1 checks all optimal tours that satisfy (i) in Proposition 11. The algorithm for the tree with endpoints v_i, v_{i+1} finds an optimal tour that starts at s, ends at t, and does not traverse $e = (v_i, v_{i+1})$.

Step 2 checks all the optimal tours of Form (1) in Theorem 12 as displayed in Fig. 1. The tour goes along B from s to v doing only deliveries, which will increase the available capacity of the vehicle. Since the s–v subpath of B will be traversed again, all pickups in that subpath will be done in the last visit therein. Thus, the tour returns from v idly to s along B. Since each edge in A is traversed twice, all the pickups in A will be done in the next visit in A, and in the first complete visit in A only deliveries are performed. Then, the tour goes from t to s along B using the parametric path algorithm. If an optimal tour is obtained for some capacity $c + S_{vu}$, it follows

from Lemma 10 that the same tour can be used for a problem where the pickups on B between t and u are not performed during this visit. The tour continues from s to t along A doing only pickups. If $u \neq t$, the tour continues along B idly to u, and returns back to t doing pickups.

Step 3 checks all the optimal tours of Form (2) in Theorem 12, as displayed in Fig. 2. The tour goes along A from s to t doing only deliveries. Since A will be traversed again, all pickups will be done in the last visit therein. Then, the tour goes along B from t to v doing only deliveries, which will increase the available capacity of the vehicle. Since this subpath will be traversed again, all pickups therein will be done in the last visit. The tour returns from v idly to t along B. It then goes along A from t to s, continuing along B from s to t, using the parametric path algorithm. If an optimal tour is obtained for some capacity $c+S_{vu}$, it follows from Lemma 10 that the same tour can be used for a problem where the pickups on A between t and u are not performed during this visit. If $u \neq t$, the tour continues along A idly to u, and returns back to t doing pickups.

Step 4 checks all the optimal tours of Form (3) in Theorem 12, as displayed in Fig. 3. The tour goes along A from s to v doing only deliveries, and returns idly to s. Since this subpath will be traversed again, all pickups will be done in the last visit therein. The parametric path algorithm is used for traversing B from s to t, and therefrom the tour goes along A from t to s doing deliveries. If the optimal tour obtained by the parametric algorithm on B is attained for an initial capacity $c + S_{vu}$, it follows from Lemma 10 that the same tour can be used for a problem where the pickups on B between s and u are not performed during the complete visit in B. Thus, if $u \neq s$, the tour continues from s along B idly to u, and returns back to s doing pickups. Then it returns along A to t doing pickups, and continues along B idly to w and back to t if any pickups were left by the parametric algorithm on B.

Step 5 checks all the optimal tours of Form (4) in Theorem 12, as displayed in Fig. 4. The tour goes from s to v along A doing only deliveries. The tour returns from v to s along A performing pickups. The parametric path algorithm is used for traversing B from s to t, and therefrom the tour then goes idly along A from t to u doing deliveries, and returns back to t doing pickups. If an optimal tour is obtained by the parametric algorithm for some capacity $c + S_{vu}$, it follows from Lemma 10 that the same tour can be used for a problem where the pickups on A between v and u are not performed during the first visit in A. It follows from Lemma 10 that only cases when $c + S_{vu}$ is non-negative have to be considered. If there were any pickups left from the path algorithm on B, the tour continues along B idly to w and back to t performing the remaining pickups.

Now, as previously noted, every tour that visits all vertices must do it in one of the ways described in Proposition 11. Further, an optimal tour of Form (ii) in Proposition 11, has one of the four structures described in Theorem 12. Therefore, the shortest of the five tours given above is an optimal tour for the cycle. □

Example. Consider the VRPD problem on a cycle graph shown in Fig. 5.

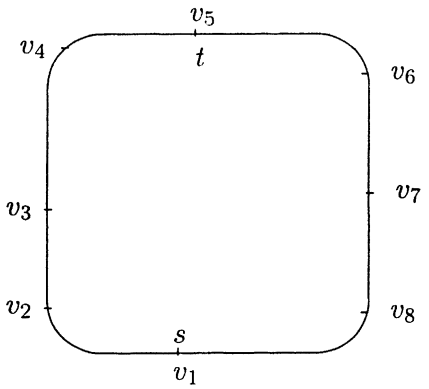

Fig. 5. Example.

The demands in the vertices are given by the following table:

v_i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$r(v_i)$	2	-2	3	-4	-1	1	-3	4

The lengths of the arcs are given by the following table:

e	(v_1,v_2)	(v_2,v_3)	(v_3,v_4)	(v_4,v_5)	(v_5,v_6)	(v_6,v_7)	(v_7,v_8)	(v_8,v_1)
$l(e)$	3	2	4	3	4	3	3	4

Now, applying the algorithm described above will result with the following.

Step 1: The first column represents the edge of the cycle that is not traversed, the second is the corresponding shortest tour, and the third is the length of the tour:

e	T	$l(T)$
(v_1,v_2)	$v_1v_8v_7v_6v_5v_4v_3v_2v_3v_4v_5v_6v_7v_8v_1v_8v_7v_6v_5$	60
(v_2,v_3)	$v_1v_2v_1v_8v_7v_6v_5v_4v_3v_4v_5v_6v_7v_8v_7v_6v_5$	54
(v_3,v_4)	$v_1v_8v_7v_6v_5v_4v_5v_6v_7v_8v_1v_2v_3v_2v_1v_8v_7v_6v_5$	58
(v_4,v_5)	$v_1v_2v_3v_4v_3v_2v_1v_8v_7v_8v_7v_6v_5v_6v_5$	46
(v_5,v_6)	$v_1v_2v_3v_4v_5v_4v_3v_2v_1v_8v_7v_6v_7v_8v_1v_2v_3v_4v_5$	56
(v_6,v_7)	$v_1v_2v_3v_4v_3v_2v_1v_8v_7v_8v_1v_2v_3v_4v_5v_6v_5$	52
(v_7,v_8)	$v_1v_2v_3v_4v_5v_6v_7v_6v_5v_4v_3v_2v_1v_8v_1v_2v_3v_4v_5$	58
(v_8,v_1)	$v_1v_2v_1v_2v_3v_4v_3v_4v_5v_6v_7v_8v_7v_6v_5$	46

$w(T_1)=46$.

Step 2 The first column represents the demand vertex where the first visit in B ends (see Fig. 1), the second is the corresponding shortest tour, and the third is the length

of the tour:

v	T	$l(T)$
v_2	$v_1v_2v_1v_8v_7v_6v_5v_4v_3v_2v_1v_8v_7v_6v_5$	46
v_4	$v_1v_2v_3v_4v_3v_2v_1v_8v_7v_6v_5v_4v_3v_2v_1v_8v_7v_6v_5$	58
v_7	$v_1v_8v_7v_8v_1v_2v_3v_4v_5v_6v_7v_8v_1v_2v_3v_4v_5$	52

$w(T_2) = 46$.

Step 3: The first column represents the demand vertex where the first visit in B ends (see Fig. 2), the second is the corresponding shortest tour, and the third is the length of the tour:

v	T	$l(T)$
v_2	$v_1v_8v_7v_6v_5v_4v_3v_2v_3v_4v_5v_6v_7v_8v_1v_2v_3v_4v_5$	58
v_4	$v_1v_8v_7v_6v_5v_4v_5v_6v_7v_8v_1v_2v_3v_4v_5$	46
v_5	$v_1v_8v_7v_6v_5v_6v_7v_8v_1v_2v_3v_4v_5v_6v_7v_8v_7v_6v_5$	60
v_5	$v_1v_2v_3v_4v_5v_4v_3v_2v_1v_8v_7v_8v_7v_6v_5$	44
v_7	$v_1v_2v_3v_4v_5v_6v_7v_6v_5v_4v_3v_2v_1v_8v_7v_6v_5$	52

$w(T_3) = 44$.

Step 4: The first column represents the demand vertex where the first visit in A ends (see Fig. 3), the second is the corresponding shortest tour, and the third is the length of the tour:

v	T	$l(T)$
v_2	$v_1v_2v_1v_8v_7v_8v_7v_6v_5v_4v_3v_2v_1v_2v_3v_4v_5$	50
v_4	$v_1v_2v_3v_4v_3v_2v_1v_8v_7v_6v_5v_4v_3v_2v_1v_2v_3v_4v_5$	56
v_7	$v_1v_8v_7v_8v_1v_2v_3v_4v_5v_6v_7v_8v_1v_8v_7v_6v_5$	54

$w(T_4) = 50$.

Step 5: The first column represents the demand vertex where the first visit in A ends (see Fig. 4), the second is the corresponding shortest tour, and the third is the length of the tour:

v	T	$l(T)$
v_4	$v_1v_2v_3v_4v_3v_2v_1v_8v_7v_6v_5v_4v_3v_4v_5$	46
v_7	$v_1v_8v_7v_8v_1v_2v_3v_4v_5v_6v_7v_8v_7v_6v_5$	46

$w(T_5) = 46$.

A shortest tour, T corresponds to T_3, with length $w(T) = 44$. It starts by performing deliveries at v_2, v_4 and v_5, and pickups at v_3 and v_1 on the way back to v_1. Continuing

its way to v_5 along the other s–t path, the tour then skips the pickup at vertex v_8, makes a delivery at v_7, returns for a pickup at v_8, and then reverses direction and makes a pickup at v_6 on its way to vertex t.

5.2.2. *The case where* $s = t$

The case where $s = t$ can be transformed to the case where $s \neq t$, by inserting a zero-length edge between s and t. Then, the algorithm for a cycle with $s \neq t$ can be applied to this case as well. However, when $s = t$, i.e. when one of the s–t paths in C is of length zero, the algorithm can be simplified. For completeness, we briefly describe the simplified algorithm. Denote by C_1 the cycle obtained from C by inserting the vertex t between s and $v_{|V|}$, with $d(t, v_{|V|}) = d(s, v_{|V|})$, and $d(s,t) = 0$, and by C_2 the cycle obtained from C by inserting the vertex t between s and v_1, with $d(t, v_1) = d(s, v_1)$, and $d(s,t) = 0$. Apply Step 1 of the algorithm for a cycle with $s \neq t$, for one of the cycles, either C_1 or C_2, and Step 2 for both cycles, C_1 and C_2. The shortest tour among those obtained in Step 1, for either C_1 or C_2, and those obtained in Step 2 for C_1 and C_2 is an optimal tour for C with $s = t$.

Implementation and correctness of the algorithm: Step 1 checks all the tours on C in which one edge is not traversed. The algorithm for a tree decides if the tour starts its first visit clockwise or counterclockwise, so we do not need to check both C_1 and C_2.

Step 2 checks all the ways of traversing the whole cycle C in both directions. On one of the cycles the optimal path algorithm is performed clockwise, and on the other counterclockwise, with the edge (s,t) of length 0 traversed two times. A tour starts (eventually) along the cycle counterclockwise (resp. clockwise) doing deliveries, returns idly clockwise (resp. counterclockwise) to s, goes along A clockwise (resp. counterclockwise) to s using the parametric path algorithm. If an optimal tour is obtained for some capacity $c + S_{vu}$, it follows from Lemma 10 that the same tour can be used for a problem where the pickups between s and u are not performed during this visit. So, if some of the pickups were left behind, the tour continues idly clockwise (resp. counterclockwise) to u and returns back to s counterclockwise (resp. clockwise) while performing the pickups.

6. The VRPD problem on a warehouse graph

We develop in this section an algorithm for solving the VRPD problem on a warehouse graph G, consisting of two parallel corridors A and B which are connected by k aisles, $k \geqslant 2$. Without loss of generality, we will assume that G is biconnected. That is, the extreme points of the parallel corridors are connected by aisles. The starting and terminal vertices for the vehicle are located on the corridors. The algorithm we develop is a recursive enumeration method, which is based on Corollary 3 and Proposition 4.

Indeed, it follows from Proposition 4 that any optimal tour, T, for a VRPD(G,r,N) problem whose first pickup is at $x_j,(x_j,b_j)\in T$, can be divided into two subtours T_{x_j} and $T\setminus T_{x_j}$, where the first performs only deliveries except one pickup in x_j, and the later is an optimal tour for the related VRPD$(G\setminus x_j, r_{T_{x_j}}, N_{T_{x_j}})$ problem[6] from x_{j+1} to t. The algorithm we develop below for a VRPD problem on a warehouse graph for a predetermined s and t, examines exhaustively all the tours that can be obtained by combining two such subtours. That is, for every supply vertex x_i in the warehouse graph, it first considers all possible subtours T_{x_i} from s to x_i that perform only deliveries until a pickup is performed in x_i. Each such subtour, T_{x_i}, is then followed by an optimal tour for the related VRPD problem. If x_j is located on one of the corridors A or B, the related problem is defined on the same graph in which vertex x_j ceases to be either a demand or a supply vertex. If x_j is an interior vertex in one of the aisles, say i, then it is shown that the corresponding related VRPD problem one needs to solve can be reduced to several VRPD problems on a warehouse graph in which aisle i is replaced by at most two supply vertices and one demand vertex, located on the two corridors A and B.

Thus, the enumeration algorithm eventually reduces the VRPD problem defined on a warehouse graph to related VRPD problems defined on cycle graphs or on warehouse graphs in which all vertices are either demand vertices or are all supply vertices. In the first case, we can use the algorithms developed for a cycle graph, while in the latter case our algorithm can use the linear time algorithm for warehouse graphs developed by Ratliff and Rosenthal [11].

We first introduce some notation. We denote by $G(k_A,k_B,k)=(V,E)$ a *generalized warehouse* graph, with k vertical aisles, connecting two horizontal corridors A and B, and k_A and k_B supply and demand vertices $a_1,\ldots,a_{k_A}$ and $b_1,\ldots,b_{k_B}$, on A and B, respectively. We allow other vertices on A and B which are neither supply nor demand vertices. Denote by P_i, $1\leqslant i\leqslant k$, the ith aisle connecting A and B and by a_{j_i} and b_{j_i} the vertices representing the intersection points between aisle P_i and corridors A and B, respectively. For simplicity, it is assumed that the vertices representing the intersection points between the corridors and the aisles are neither supply nor demand vertices. Observe that a simple warehouse graph with k aisles, studied by Ratliff and Rosenthal [11], is obtained from $G(k_A,k_B,k)$ by setting $k_A=k_B=0$ and a cycle and paths are obtained as special cases of $G(k_A,k_B,k)$ when $k=2$ and 1, respectively. In this section we analyze the VRPD$(G(k_A,k_B,k),r,N)$ problem with $s,t\in\{A\cup B\}$, and develop an algorithm for finding an optimal tour which is polynomial in the number of supply/demand vertices for fixed k_A, k_B and k.

Our algorithm is a recursive enumeration method which solves the VRPD problem on $G(k_A,k_B,k)$ by reducing it to VRPD problems on either a $G(k_A-1,k_B,k)$ graph or a $G(k_A,k_B-1,k)$ graph, or a graph derived from $G(k_A,k_B,k)$ by disconnecting one of the aisles therein, say P_i, into two line segments, one connected to A via vertex a_i and the other connected to B via vertex b_i. Such a graph is denoted by $G(k_A,k_B,k-1)_{a_i,b_i}$. The

[6] For definition of T_{x_j} and the related VRPD problem see Section 2.

algorithm for a VRPD problem on a $G(k_A, k_B, k-1)_{a_i,b_i}$ graph, developed in Section 6.1, reduces it to VRPD problems defined on $G(k_A+q, k_B+p, k-1)$ graphs with $p,q \in \{1,2\}$, or on a $G(k_A, k_B, k-1)$ graph. It is important to point out that even though k_A and k_B are increased in the recursive calls, the overall increase until the end of the recursion is bounded by $2k$. Indeed, both k_A and k_B are increased by at most two per each aisle, during the execution of the entire algorithm. The recursion ends either with a VRPD problem on a cycle graph, or with a graph which does not contain any supply vertices.

An algorithm for the VRPD$(G(k_A, k_B, k), r, N)$ *problem*

1. If $k=2$ then $G(k_A, k_B, k)$ forms a cycle graph. Solve the problem using the algorithm developed in Section 5.2 for cycle graphs.
2. If there is no supply vertex in $G(k_A, k_B, k)$ or if there is no demand vertex therein, find a shortest travelling salesman tour starting at s and ending at t using, for example, the algorithm developed by Ratliff and Rosenthal [11].
3. Consider all connected subgraphs G_S of G containing s and at least one supply vertex. It follows from the connectivity of G_S that for every aisle P_i in G, either P_i is contained in G_S, or there exist vertices x_i and y_i on P_i, whose degrees are one in G_S, such that the subpaths a_{j_i}–x_i and y_i–b_{j_i} of P_i are contained in G_S. Let S be the set which consists of all demand vertices in G_S. For every such set S and a supply vertex $v \in G_S$ do:
   (i) Compute a minimum length tour $T_{S,v}$, which traverses G_S by starting from s, ending at v and performing only all the deliveries in G_S, except for a pickup at v at the end of the tour. $T_{S,v}$ is a shortest Hamiltonian path in G_S from s to v, with respect to $S \cup \{s, v\}$, and can be found in linear time.
   (ii) If v is an interior vertex on aisle P_i, consider VRPD$(G_{T_{S,v}}, r_{T_{S,v}}, N_{T_{S,v}})$, the related VRPD problem, where[7] $G_{T_{S,v}} = G \setminus v = G(k_A, k_B, k-1)_{a_i,b_i}$, $r_{T_{S,v}}$ are the new requirements derived by accounting for deliveries made prior to reaching v, and pickup at v, made in the subtour $T_{S,v}$, and $N_{T_{S,v}}$ is the modified capacity of the vehicle. Find recursively optimal tours, T_1 and T_2, for VRPD$(G(k_A, k_B, k-1)_{a_{j_i},b_{j_i}}, r_{T_{S,v}}, N_{T_{S,v}})$ that start at one of the two vertices, v' and v'', adjacent to v in $G(k_A, k_B, k)$, respectively, and end in t, using the algorithm developed in Section 6.1 below. An optimal tour, $T^a(S, v)$, for the VRPD problem on $G(k_A, k_B, k)$, in which deliveries are done at S before the first pickup at v, is the one with corresponding shortest length between the two tours obtained as a concatenation of $T_{s,v}$ with T_1 and T_2, respectively.
   (iii) If v is not an interior vertex of one of the aisles assume, without loss of generality, that $v \in A$. Then, collect all the supply at v, and the resulting related graph, $G(k_A-1, k_B, k)$, coincides with $G(k_A, k_B, k)$ with the only exception that v has no requirements. Let T_b denote an optimal tour for the related VRPD problem,

[7] If P_i is an extreme aisle, with an adjacent aisle, say P_j, then in $G(k_A, k_B, k-1)_{a_i,b_i}$, we set $a_i := a_j, b_i := b_j$, and the line segments attached to a_i and b_i consist of segments of aisle P_i and the subpaths a_i–a_j and b_i–b_j, on A and B, respectively.

VRPD($G(k_A-1,k_B,k), r_{T_{S,v}}, N_{T_{S,v}}$), defined on $G(k_A-1,k_B,k)$ which starts at v and ends at t, and in which the requirements, $r_{T_{S,v}}$, are derived by accounting for deliveries and pickup at v made in $T_{S,v}$, and the vehicle capacity, $N_{T_{S,v}}$, is modified accordingly. Then, an optimal tour, $T^b(S,v)$, for the VRPD problem on $G(k_A,k_B,k)$, in which deliveries are made at S before a first pickup at v, is the concatenation of $T_{S,v}$ and T_b.

4. Let $\hat{T}^a(S_0,v_0)$ denote a tour for which $w(\hat{T}^a(S_0,v_0)):=\min\{T^a(S,v)$: all S satisfying the above requirement and v an interior aisle vertex$\}$ and let $\hat{T}^b(S_1,v_1)$ denote a tour for which $w(\hat{T}^b(S_1,v_1)):=\min\{w(T^b(S,v)$: all S satisfying the above requirements and v not an interior aisle vertex.$\}$
5. If $w(\hat{T}^a(S_0,v_0)) < w(\hat{T}^b(S_1,v_1))$, then $\hat{T}^a(S_0,v_0)$ is an optimal tour for the VRPD problem on $G(k_A,k_B,k)$. Otherwise, $\hat{T}^b(S_1,v_1)$ is an optimal tour for $G(k_A,k_B,k)$.

Let us denote by $t(G(k_A,k_B,k))$ the running time of the algorithm on $G(k_A,k_B,k)$ and by V_A, V_B, and V_P, the supply vertices on A, B, and the aisles, respectively. Then

$$\begin{aligned} t(G(k_A,k_B,k)) = {} & \mathrm{O}(|V|^{2k})\mathrm{O}(|V|)\cdot \\ & [\mathrm{O}(|V_P|)2t(G(k_A,k_B,k-1)_{a,b}) + \mathrm{O}(|V_A|)t(G(k_A-1,k_B,k)) \\ & + \mathrm{O}(|V_B|)t(G(k_A,k_B-1,k))]. \end{aligned}$$

Indeed, there are $\mathrm{O}(|V|^{2k})$ subsets S for which Step 3 of the algorithm is performed. For each such set S, a minimum length tour, $T_{S,v}$ can be computed in $\mathrm{O}(|V|)$ operations. If the first pickup is at an aisle, the problem on $G(k_A,k_B,k)$ is reduced to two problems defined on $G(k_A,k_B,k-1)_{a,b}$ graphs. If the first pickup is on the corridors then the problem is reduced to one defined on the same topological graph with one fewer supply vertex on the corridors.

6.1. The VRPD problem on a $G(k_A,k_B,k)_{a_i,b_i}$ graph

In this subsection we develop an algorithm for a VRPD problem on a $G(k_A,k_B,k)_{a_i,b_i}$ graph, with s being an end point with degree one of one of the line segments connected to a_{j_i} and b_{j_i} and t is in A or B. For simplicity of presentation, we denote the vertices a_{j_i} and b_{j_i} by a and b, and the line segments connected to them by P_a and P_b, respectively. Without loss of generality, we assume that $s=v_a$, the end point of P_a. Note that P_a may consist only of a, in which case v_a coincides with a. Similarly, v_b, the end point of P_b, may coincide with b.

The algorithm is based on the following two lemmas.

Lemma 20. *An optimal tour T for $G(k_A,k_B,k)_{a,b}$ that starts at v_a, and ends at $t\in\{A,B\}$, performs the following* before *it* first *reaches a. It goes along some v_a–v subpath (maybe empty) of P_a, as an optimal tour along the line segment v_a–v, then it continues to a, performing only deliveries if any.*

Proof. To see this, let v be the closest supply vertex to a on P_a at which a pickup is performed by T before it reaches a for the first time. It follows from Corollary 3 that all the requests in the subpath v_a–v are fulfilled before the pickup at v is performed. The optimality of T implies that the subtour of T on v_a–v is also optimal. Since v was chosen as the closest vertex to a at which a pickup was performed, then after the pickup at v the tour performs only deliveries in the v–a subpath. □

Lemma 21. *P_a, P_b are visited by an optimal tour T that starts in v_a at most twice.*

Proof. Denote by $P \in \{P_a, P_b\}$ the line segment visited by T more than twice. If $P = P_a$, then by Lemma 20, after the first visit that starts at v_a and leaves from a, only pickup requests are left in P_a. Obviously, all these pickup requests can be performed when T last visits vertex a.

If $P = P_b$, we show how to construct a shorter tour T_1 which visits P at most twice, contradicting the optimality of T. Suppose T visited P i times, $i > 2$. Each time T visited P, it performed some of the requests along P. Denote by $h_1, \ldots, h_i$ the sum of the requests performed by T in each of the visits of T in P, and by $l_1, \ldots, l_i$ the lengths of the subtours of these visits, respectively. Partition the set of all the visits of T in P into two sets as follows: T_N, the set of all such visits for which h_j is negative, and T_P, the set of all such visits for which h_j is positive. For $i > 2$, we can derive from T a shorter tour, T_1, as follows. T_1 follows T until T's first visit to P. When T_1 first visits P, it performs all the requests performed by T in all its visits at P that belong to T_N. Then, T_1 leaves P and continues as T, skipping T's remaining visits at P, except the last one. At this point, T_1 enters P, performing all the requests done by all of T's visits at P that belong to T_P. It essentially follows from Observation 1 that T_1 is a feasible tour. Further, observe that T_1 contains just one visit to P instead of all of T's visits at T_N and one visit instead of all of T's visits at T_P. The length of these two visits at P by T_1 are equal to the lengths of the longest visit in T_N and the longest visit in T_P to P by T, respectively. It follows that if $i > 2$ then $l(T_1) < l(T)$ and the lemma follows. □

Let T be an optimal tour. Then, since T visits P_b at most twice we have the following corollary:

Corollary 22. *There exists a vertex $u \in P_b$ such that the two visits, T_1 and T_2 of T to P_b, fall in one of the following two categories*:

(a) *The first visit to P_b involves entering from b, delivering in the b–u subpath and returning idly. The second visit involves entering from b going idly to u, delivering from u to v_b and picking up on the way back from v_b to b. Note that the first visit may be empty, and both visits are empty if P_b consists merely of b.*

(b) *The first visit to P_b involves entering from b, delivering along the entire segment P_b while picking up on the way back from v_b to u (except in u), and continuing idly back to b. The second visit involves entering from b going idly to u, and*

picking up on the way back only from u to b. Note that the second visit may be empty, and both visits are empty if P_b consists of b.

In both categories, the total length of these two visits is $l(T_1) + l(T_2) = 2l(P_b) + 2l(b-u)$.

Lemmas 20 and 21 and Corollary 22 suggest a reduction, for each pair of vertices $v \in P_a$ and $u \in P_b$, of the VRPD problem on a $G(k_A, k_B, k)_{a,b}$ graph to a pair of VRPD problems on graphs $G(k_A + p, k_B + q, k)$, where $p \leqslant 1$ and[8] $q \leqslant 2$. The new supply or demand vertices are inserted on the corridors A and B to represent the total load picked up and or delivered by any feasible tour in $G(k_A, k_B, k)_{a,b}$ during its excursions into P_a and P_b. The nature of these excursions is determined by $v \in P_a$ and $u \in P_b$. Explicitly, each pair of vertices $v \in P_a$ and $u \in P_b$ induce two possible sets of visits to P_a and P_b by any feasible tour, where each set involves one visit entering from a to P_a and two visits entering from b to P_b. For each such set of visits, we modify the requirement at vertices a and b and[9] insert an additional vertex, b', which is placed on B at a zero distance from b. The length of all other edges in $G(k_A, k_B, k)_{a,b}$, excluding P_a and P_b, does not change and for any vertex q, $l(q, b') = l(q, b)$. The request functions, r_1 and r_2, associated with the pair of VRPD problems on the pair of $G(k_A + p, k_B + q, k)$ graphs are defined as follows:

- $r_1(a) = r_2(a) := \sum(r(w)$: w is a supply vertex in the v–a subpath of $P_a)$.
- $r_1(b) := \sum(r(w)$: w is a demand vertex along the b–u subpath$)$.
- $r_1(b') := \sum(r(w)$: w is a demand vertex along the u–v_b subpath$) + \sum(r(w)$: w is a supply vertex along $P_b)$.
- $r_2(b) := \sum(r(w)$: w is a demand vertex along $P_b) + \sum(r(w)$: w is a supply vertex along the v_b–u subpath$)$.
- $r_2(b') := \sum(r(w)$: w is a supply vertex along the u–b subpath$)$.

Both r_1 and r_2 coincide with r on all vertices other than a, b and b'.

We are now ready to present the algorithm for a VRPD problem defined over a $G(k_A, k_B, k)_{a,b}$ graph.

An algorithm for a VRPD $(G(k_A, k_B, k)_{a,b}, r, N)$ problem with $s = v_a$ and $t \in A \cup B$
Step 1: For every pair of vertices, $v \in P_a$ and $u \in P_b$, do:

- Perform an optimal tour along the v_a–v subpath from v_a to v, then traverse the v–a subpath performing only deliveries along the way. Denote this subtour by $T_a(v)$ and by N' the related vehicle after performing $T_a(v)$.
- Construct the related VRPD problems, VRPD$(G(k_A + 1, k_B + 2, k), r_1, N')$ and VRPD$(G(k_A + 1, k_B + 2, k), r_2, N')$.

[8] $p = 1$ if there are supply vertices, which are interior vertices of the subpath v–a of P_a. Otherwise, $p = 0$. Similarly, $q = 0, 1$ or 2 if there are no visits at P_b, one visit to P_b, or two visits at P_b, respectively.

[9] Observe that before this modification, vertices a and b are neither supply nor demand vertices in $G(k_A, k_B, k)_{a,b}$.

- Find an optimal tour, $T_1(v,u)$, for the VRPD$(G(k_A+1,k_B+2,k),r_1,N')$ problem from a to t.
- Find an optimal tour, $T_2(v,u)$, for the VRPD$(G(k_A,k_B+1,k),r_2,N')$ problem from a to t.
- Choose $T(v,u)$ as $T_a(v)$ followed by the minimal of $T_1(v,u)$ and $T_2(v,u)$. $w(T(v,u)):=w(T_a(v))+\min\{w(T_1(v,u)),w(T_2(v,u))\}+2l(a-\hat{v})+2l(P_b)+2l(b-u)$, where $\hat{v}\neq v$ is a supply vertex on the subpath $a-v$ closest to v. If no such vertex $\hat{v}$ exists, then $\hat{v}=a$.

Step 2: $w(T(v_0,u_0)):=\min_{v,u}\{(w(T(v,u)))\}$ and $T(v_0,u_0)$, with the corresponding excursions into P_a and P_b, is an optimal tour for the VRPD problem on $G(k_A,k_B,k)_{a,b}$.

To calculate the time complexity of our algorithm, recall that

$$\begin{aligned} t(G(k_A,k_B,k)) = {} & \mathrm{O}(|V|^{2k})\mathrm{O}(|V|)\cdot \\ & [\mathrm{O}(|V_P|)2t(G(k_A,k_B,k-1)_{a,b})+\mathrm{O}(|V_A|)t(G(k_A-1,k_B,k)) \\ & +\mathrm{O}(|V_B|)t(G(k_A,k_B-1,k))]. \end{aligned}$$

Therefore, to complete the analysis of the algorithm for the VRPD$(G(k_A,k_B,k),r,N)$ problem, we need to determine the time complexity of the algorithm on a $G(k_A,k_B,k)_{a,b}$ graph. Since there are $\mathrm{O}(|V|^2)$ pairs of vertices (v,w) for which Step 1 is performed, the time complexity of the algorithm on a $G(k_A,k_B,k)_{a,b}$ graph starting from v_a is

$$t(G(k_A,k_B,k)_{a,b}) = \mathrm{O}(|V|^2)2t(G(k_A+1,k_B+2,k))$$

and the time complexity of the algorithm on a $G(k_A,k_B,k)_{a,b}$ graph starting from v_b is

$$t(G(k_A,k_B,k)_{a,b}) = \mathrm{O}(|V|^2)2t(G(k_A+2,k_B+1,k)).$$

Assume that we apply the algorithm to a warehouse graph, $G(k_A,k_B,k)$. Then, throughout the execution of the whole algorithm there are at most $k-2$ recursive calls to the algorithm in Section 6.1, since each such recursive call reduces the number of aisles by one. On the other hand, each such recursive call increases k_A+k_B by at most three. Thus, until the end of the recursion, k_A+k_B may be increased by at most $3(k-2)$. There are at most $2(k-2)+k_A+k_B$ recursive [10] calls of the algorithm to itself, since each call reduces either k_A or k_B by one. We get that

$$\begin{aligned} t(G(k_A,k_B,k)) &\leqslant \mathrm{O}(|V|^{2k+4})^{k-2}\mathrm{O}(|V|^{2k+2})^{2(k-2)+k_A+k_B} \\ &= \mathrm{O}(|V|)^{k^2}\mathrm{O}(|V|)^{k(k_A+k_B+k)}. \end{aligned}$$

The time complexity is exponential in k and k_A+k_B, but polynomial when k and k_A+k_B are fixed. That is, the algorithm is polynomial in the number of pickup and delivery vertices on the aisles, when the number of aisles, k, and the number of demand and supply vertices on the corridors A and B, k_A+k_B, are not part of the problem input. □

[10] Recall that $k-2$ out of the $3(k-2)$ new vertices inserted on the corridors, as a result of the elimination of $k-2$ aisles, are demand vertices.

References

[1] S. Anily, R. Hassin, The swapping problem, Networks 22 (1992) 695–703.
[2] S. Anily, G. Mosheiov, The traveling salesman problem with delivery and backhauls, Oper. Res. Lett. 16 (1994) 11–18.
[3] L. Bodin, B. Golden, A. Assad, M. Ball, Routing and scheduling of vehicles and crews – state of the art, Comput. Oper. Res. 10 (1983) 63–211.
[4] D. Casco, B. Golden, E. Wasil, Vehicle routing with backhauls: models, algorithms and case-studies, in: B. Golden, A. Assad (Eds.), Vehicle Routing: Methods and Studies, Studies in Management Science and Systems, Vol. 16, North-Holland Publishing, Amsterdam, 1988.
[5] P. Chalasani, R. Motwani, Approximating capacitated routing and delivery problems, SIAM J. Comput. 28 (1999) 2133–2149.
[6] P. Chalasani, R. Motwani, A.S. Rao, Algorithms for robot grasp and delivery, Proceedings of 2nd International Workshop of Algorithmic Foundations of Robotics, Toulouse, France, July 1996, pp. 347–362.
[7] D.J. Guan, Routing of a vehicle of capacity greater than one, Discrete Appl. Math. 81 (1998) 41–57.
[8] G. Mosheiov, The traveling salesman problem with pickup and delivery, European J. Oper. Res. 79 (2) (1994) 299–310.
[9] G. Mosheiov, The pickup delivery location problem on networks, Networks 26 (1995) 243–251.
[10] G. Mosheiov, Vehicle routing with pick-up and delivery: tour-partitioning heuristics, Working Paper, The School of Business Administration and Department of Statistics, Hebrew University, Jerusalem, Israel, 1996.
[11] H.D. Ratliff, A.S. Rosenthal, Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem, Oper. Res. 31 (3) (1983) 507–521.
[12] C. Yano, T. Chan, L. Richter, T. Cutler, K. Murty, D. McGettigan, Vehicle routing at quality stores, Interfaces 17 (1987) 52–63.